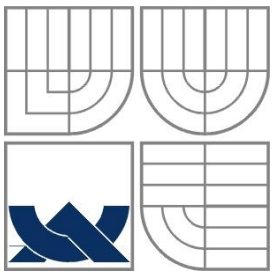
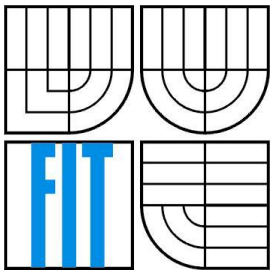


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# WEBOVÉ ROZHRANÍ SYSTÉMU PRO ANALÝZU DOKUMENTŮ

WEB INTERFACE FOR A DOCUMENT ANALYSIS SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

OLGA MIRSKÁ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. RADEK BURGET, PH.D.

BRNO 2016

# Abstrakt

Tato bakalářská práce popisuje návrh a realizaci webového rozhraní systému pro analýzu internetových dokumentů na základě existující desktopové aplikace FITLayout Framework. Práce je rozdělena na teoretickou a praktickou část. V teoretické části jsou popsány použité technologie pro tvorbu dynamických webových stránek. Praktická část se zabývá návrhem řešení a popisem implementace jednotlivých funkcí webové aplikace FITLayout. V závěru jsou uvedeny výsledky testování funkcí webové aplikace a jejich porovnání s funkcemi FITLayout Framework.



# Abstract

This Bachelor's thesis describes the design and implementation of a web interface for a internet documents analysis system based on existing desktop application FITLayout Framework. The work is divided into theoretical and practical part. The theoretical part describes the technologies used for creating dynamic web pages. The practical part is dedicated to description of design solution and implementation of particular functions of the web application FITLayout. In conclusion the testing results of the web application's functions and their comparation with functions of FITLayout Framework are presented.



# Klíčová slova

FITLayout Framework, HTML, CSS, Bootstrap, JavaScript, jQuery, Java EE, Eclipse, Apache Tomcat, dynamická webová stránka, webová aplikace

# Keywords

FITLayout Framework, HTML, CSS, Bootstrap, JavaScript, jQuery, Java EE, Eclipse, Apache Tomcat, dynamic web page, web application


# Citace

Mirská Olga: Webové rozhraní systému pro analýzu dokumentů, bakalářská práce, Brno, FIT VUT v Brně, 2016

# Webové rozhraní systému pro analýzu dokumentů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci na téma Webové rozhraní systému pro analýzu dokumentů vypracovala samostatně pod vedením Ing. Radka Burgeta, Ph.D.
Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……
Olga Mirská
16. května 2016

## Poděkování

Ráda bych poděkovala svému vedoucímu práce Ing. Radkovi Burgetovi, Ph.D. za věnovaný čas a odbornou pomoc při vypracovávání mé bakalářské práce.

# Obsah

# 1. Úvod

Cílem bakalářské práce je navržení a implementace dynamické webové aplikace, která slouží pro analýzu segmentace webových stránek. Navrhovaný program se jmenuje webová aplikace FITLayout. Program je navrhován na základě Zadání pro bakalářskou práci "Webové rozhraní systému pro analýzu dokumentů" a manuálu k softwaru FITLayout Web Page Analysis Framework. [1]

## 1.1 Popis problematiky a cíl práce

Na FIT VUT Brno je vyvíjeno rozhraní FITLayout Framework pro segmentaci a analýzu webových stránek. Toto rozhraní je implementováno na platformě Java a v současné době disponuje grafickým uživatelským rozhraním, které využívá standardní grafickou sadu nástrojů Java Swing. Tyto nástroje je možné spouštět pouze z lokálního disku a nemají webové rozhraní, což znesnadňuje běžnému uživateli jejich používání. Cílem práce je proto implementovat webovou aplikaci poskytující podmnožinu funkcí dostupných v existujících nástrojích. Řešení je rozděleno do následujících kroků:

- **nastudování existujících informací k problematice**
  - nastudování existujících nástrojů pro tvorbu klientské části dynamických webových aplikací, zejména HTML5, CSS, JavaScript
  - nastudování technologií realizace serverové části aplikace s použitím jazyka Java a aplikačních serverů (Tomcat, GlassFish apod.)
  - seznamení s FITLayout Framework, jeho funkcemi a uživatelským rozhraním
- **navržení funkcí implementované aplikace**
  Navrhovaná aplikace má umožnit uživateli analyzovat rozložení dat z libovolných web stránek a ke svému spuštění má používat běžně dostupný webový prohlížeč. Seznam funkcí se zakládá na funkcích programu FITLayout Framework, proto je třeba provést analýzu funkcí FITLayout Framework a uzpůsobit jejich seznam tak, aby umožňovaly uživateli provádět základní analýzu segmentace webových stránek v dostatečném rozsahu a na druhou stranu aby jejich počet odpovídal požadovanému rozsahu bakalářské práce. Aplikace bude koncipována jako otevřená, což znamená, že v budoucnu bude možné přidávat k ní další uživatelské funkce.
- **navržení uživatelského rozhraní a způsobu implementace**
  - Uživatelské rozhraní aplikace má umožňovat zadání url adresy analyzované webové stránky, výběr nástrojů pro analýzu (Renderer, Segmentator), rozměrů stránky, ovládacích prvků, oblastí pro prohlížení a jiné. Při návrhu uživatelského rozhraní bude třeba brát v potaz odlišnost jednotlivých prohlížečů a možnost použití obrazovek s různým rozlišením, t.j. uživatelské rozhraní se musí přizpůsobovat použitému uživatelem hardwaru a softwaru. Pro implementaci bude použita základní technologie klient-server s použitím jazyků HTML5, CSS, Javascript na klientské straně a jazyku Java na serverové straně.
- **testování a dokumentování**
  - Po ukončení implementace musí být provedeno testování aplikace, které má ověřit počet a správnost vykonávání jednotlivých funkcí. Výsledky testování včetně popisu testovacích postupů budou uvedeny v kapitole 5.1.
  - Dokumentování celého procesu návrhu, implementace a testování se bude provádět průběžně a výsledky budou uvedeny v závěru.

## 1.2 Struktura práce

Tato se skládá z následujících kapitol:
1. Úvod - popisuje základní problematiku a cíle této práce.
2. Teorie - podrobně popisuje informace existujících technologiích pro tvorbu dynamických webových aplikací, které byly použity při návrhu a implementaci aplikace.
3. Návrh řešení - popisuje hlavní problémy, které bylo třeba řešit a zdůvodňuje způsob řešení a použité nástroje.
4. Implementace - vysvětluje jak byly zmíněné problémy prakticky vyřešeny a také samotný program a detaily jeho fungování.
5. Závěr - shrnuje výsledky testování a také popisuje problémy, které byly vyřešeny a které musí být dořešeny později a uvádí literaturu a jiné zdroje, které autor použil při vypracování bakalářské práce.

# 2. Související technologie

V této kapitole se nachází teoretický úvod k jednotlivým technologiím, které byly použity při vytváření webové aplikace.

## 2.1 Technologie pro tvorbu klientské části

### 2.1.1 Jazyk HTML

HTML (HyperText Markup Language) - hypertextový značkovací jazyk, který se používá k vytváření internetových stránek. Umožňuje zvýraznit v textu jednotlivé logické části (nadpisy, odstavce, seznamy atd.), umístit na webovou stránku fotografii nebo obrázek a organizovat na stránce odkazy na další dokumenty. [2]

Základním prvkem hypertextového značkovacího jazyka je tzv. tag. Všechny tagy mají stejný formát: začátek "<", jméno a konec ">". Pro tagy stejného jména obvykle existují otevírací a uzavírací tag. Rozdíl je v tom, že v uzavíracím tagu je před jménem tagu lomítko "/". Otevírací tag se používá k určení začátku HTML elementu a nastavení vlastností (atributů) elementů umístěných za jménem tagu. Uzavírací tag se používá k označení konce elementu. Mezi otevíracím a uzavíracím tagem se mohou nacházet další tagy a nebo text. Obvykle se zadávají všechny tagy malými písmeny. Například `<html>`, `<body>`, `<h1>`, `<h2>` aj. [3]
Obecná syntaxe zápisu tagů je následovná:

```
<tag atribut1="hodnota" atribut2="hodnota">
<tag atribut1="hodnota" atribut2="hodnota">...</tag>
```

Tagy mohou mít atributy s hodnotami. Atributy umožňují měnit vlastnosti elementu, pro který jsou dané a jsou vždy uvedeny v otevíracím tagu. Většina atributů jsou nepovinné a používají se pouze pro změny standardních vlastností tagu. [3]
Příklad použití atributu:

```
<div class="head">...</div>
```

Atributy se skládají z dvojice: název atributu = "hodnota". Jména atributů mohou být psána libovolnou kombinací velkých a malých písmen. Hodnotou atributu může být text, číslo nebo jiné znaky, kromě znaku ampersand (&), který není povolen. Pořadí atributů v tagu nemá vliv na zobrazení elementu. [3]

Nejběžnější atributy jsou `class` a `id`, které se používají pro identifikaci elementů. Lze je použít u všech HTML elementů, za výjimkou těch, které obsahují technické informace: `<html>`, `<head>`, `<meta>`, `<title>`, `<style>` a `<script>`. Každému elementu může být přiřazeno několik hodnot atributu `class` a pouze jedna hodnota atributu `id`. [3]

### 2.1.2 HTML dokument

HTML dokument představuje hierarchickou (stromovou) strukturu, která se skládá z tagů a popisuje webovou stránku. Může být vytvořen buď v jednoduchém textovém editoru (Poznámkový blok), nebo ve specializovaném editoru se zvýrazněním syntaxe (např. Notepad++). HTML dokumenty jsou zpracovávány webové prohlížeči a výsledkem zpracování je zobrazení webové stránky.

#### 2.1.2.1 Struktura HTML dokumentu

HTML dokument se řídí pravidly, která jsou obsažena v definici typu dokumentu (Document Type Definition, nebo DTD). DTD je dokument, který definuje, jaké tagy, atributy a jejich hodnoty platí pro určitý typ HTML dokumentu. Každá verze jazyka HTML má svůj DTD. [3]

Každý HTML dokument, který splňuje specifikace libovolné verze jazyka HTML, musí začínat deklarací verze HTML, pomocí tagu `<!DOCTYPE>`: [4]

```
<!DOCTYPE html>
```

Tento řádek pomáhá prohlížeči zjistit jak interpretovat kód webové stránky. V tomto případě říkáme prohlížeči, že HTML dokument odpovídá standardní specifikaci.

Ačkoli se slovo DOCTYPE píše do lomených závorek (< >), není to tag, ale instrukce, určená pro webové prohlížeče a vykřičník (!) na začátku ji odlišuje od zbytku kódu v HTML dokumentu. [4]

**Element `<html>`**

Po deklaraci verze a typu dokumentu je nutné definovat jeho začátek a konec. To se provádí pomocí elementu `<html>`. Tento element se nazývá kořenový, protože všechny ostatní elementy dokumentu jsou umístěny v něm. Kořenový element může mít pouze dva podřízené elementy: `<head>` a `<body>`. [3]

**Element `<head>`**

Element `<head>` obsahuje technické informace o stránce: název, popis, klíčová slova pro vyhledávače, kódování atd. Zadané informace se nezobrazují v okně prohlížeče, ale obsahují data, která říkají prohlížeči, jak zpracovat stránku. Element `<head>` musí být prvním potomkem `<html>`. Žádné jiné elementy nesmí být umístěny před ním. Uvnitř elementu `<head>` se nachází další elementy: [3]

- **Element `<title>`**
  Povinným tagem elementu `<head>` je tag `<title>`. Text umístěný uvnitř tagu se zobrazí v záhlaví webového prohlížeče. Délka záhlaví nesmí být delší než 60 znaků. Text záhlaví by měl obsahovat popis obsahu webových stránek. [3]

- **Element `<meta>`**
  Tag `<meta>` je nepovinným tagem, který obsahuje popis obsahu stránky, klíčová slova pro vyhledávače, autora HTML dokumentů a další vlastnosti (metadata). V elementu `<head>` se může nacházet více elementů `<meta>`, podle atributů, které obsahují různé informace. [3]

- **Element `<style>`**
  V tomto elementu se nastavují styly, které jsou použity na stránce. Pro popis stylů v HTML dokumentu se používá jazyk CSS. Tento jazyk umožňuje zapisovat kód formátování jednotlivých elementů i celé webové stránky bez nutnosti měnit její HTML kód. Může být použito více elementů `<style>`. [3]

- **Element `<link>`**

Druhým způsobem jak nastavit styly pro dokument je napsat je do samostatného souboru s příponou css, například `style.css`. Připojit soubor se styly k webové stránce je možné s použitím elementu `<link>`. Tento element definuje vztah mezi aktuálním HTML dokumentem a dalšími dokumenty. Takových elementů může být na stránce několik. [3]

- **Element `<script>`**
  Element <script> umožňuje připojit k dokumentu různé skripty. Text skriptu může být umístěn buď uvnitř elementu, nebo v externím souboru. Pokud je skript umístěn v externím souboru, připojuje se pomocí atributů elementu (url adresy umístění skriptu). [3]

**Element `<body>`**

V elementu `<body>` je umístěn veškerý obsah webové stránky. V této části se nachází text, grafika, tabulky a další elementy obsahu stránky, které uživatel vidí při prohlížení webu. [3]

Příklad obecné struktury HTML souboru:

```
<!DOCTYPE html>
<html>
    <head>
        <title>...</title>
        <meta>...</meta>
        <style>...</style>
        <link>...</link>
        <script>...</script>
    </head>

    <body>
        ...
    </body>
</html>
```

### 2.1.3 HTML5

HTML5 není nástupcem hypertextového značkovacího jazyka, ale nová otevřená platforma pro vytváření webových aplikací pomocí audio, video, grafiky, animací a dalších. Vzniklo množství nových sémantických elementů a tagů, které umožňují vložit na web audio a video. [5]

**Některé funkce HTML5:**

- Podpora geolokace - určení polohy uživatele na mapě a použití této informace k výpočtu cesty jeho pohybu, nalezení okolních obchodů, kin, kaváren a dalších údajů
- Přehrávání videí
- Přehrávání zvukových souborů
- Lokální úložiště - umožňuje webovým stránkám ukládat informace na lokálním počítači a přistupovat k nim později
- Výpočet na pozadí - standardní způsob, jak spustit JavaScript v prohlížeči na pozadí
- Off-line aplikace - stránky, které mohou pracovat bez připojení k internetu
- Kreslení - uvnitř tagu <canvas> je možné s použitím JavaScriptu kreslit tvary, čáry, vytvářet přechody a transformovat objekty v reálném čase
- Nové formulářové prvky pro datum, čas, hledání, čísel, výběr barev a další [5]

### 2.1.4 Jazyk CSS

CSS (Cascading Style Sheets) je formální jazyk pro popis vzhledu dokumentu napsaného pomocí značkovacího jazyka. Hlavním smyslem kaskádových stylů je sdílet obsah webových stránek a formát (reprezentaci) dokumentu. To znamená, že v kódu HTML stránky jsou informace, které uživatel vidí po otevření této stránky v prohlížeči a to, v jaké podobě vidí tyto informace na obrazovce, vytváří soubor kaskádových stylů. Soubor kaskádových stylů je textový soubor, který má obvykle příponu css. Rozšíření css umožňuje prohlížeči identifikovat daný soubor právě jako kaskádové styly a správně interpretovat jeho obsah. Výhodou je, že pouze jeden soubor kaskádových stylů obsahuje informaci o formátování celé webové stránky. Pro připojení kaskádových stylů, musí HTML kód stránky obsahovat tento meta tag: [6]

```
<link rel="stylesheet" href="style.css" type="text/css">
```

Název souboru kaskádových stylů a cesta k němu se uvádí v atributu href.

**Konkrétní výhody CSS:**

- nastavení zobrazení mnoha dokumentů pomocí jednoho souboru kaskádových stylů
- kontrola vzhledu stránek bez nutnosti měnit jejich kód
- odlišná reprezentace pro různá média (monitor, tisk atd.)
- komplexní a propracovaná technika designu

#### 2.1.4.1 Selektory a vlastnosti

Tak jako jazyk HTML má tagy, má jazyk CSS selektory. Pomocí selektorů je možné vybrat elementy na stránce, které chceme naformátovat. Selektorem může být jméno tagu, jeho id, class nebo jejich množina. Pro každý selektor se definují vlastnosti, které jsou umístěny uvnitř složených závorek. Za jménem vlastnosti následuje dvojtečka a za ní hodnota této vlastnosti. Tyto vlastnosti platí pro všechny elementy, které odpovídají selektoru: mají stejné jméno tagu, stejné id nebo class. Více po sobě jdoucích vlastností se oddělují středníkem, například: [6]

```
body {
    font-size: 0.8em;
    color: navy;
}
```

Tento kód nastaví pro tag `<body>`, který je selektorem, hodnoty vlastností font-size a color.

Kaskádové styly umožňují nastavovat například tyto vlastnosti elementů:

- písmo
- formátovaní textu
- barva pozadí a popředí
- zobrazení ovládacího prvku
- správa seznamů
- formátování tabulek
- velikost prvků
- zobrazení prvků
- ohraničení, odsazení, pole
- správa rozvržení stránky
- správa rozhraní

### 2.1.5 Knihovna Bootstrap

Bootstrap je volně šiřitelný CSS/HTML framework pro tvorbu webových stránek. Jinými slovy, je to sada nástrojů pro formátování stránky. Má řadu výhod, díky kterým je Bootstrap považován za nejpopulárnější framework svého druhu (používaný např. ve Facebooku). [7]

**Výhody bootstrapu:**

- Rychlost práce - díky řadě hotových elementů s pomocí bootstrapu trvá rozvržení stránky kratší dobu
- Rozšiřitelnost - přidání nových elementů nenarušuje obecnou strukturu
- Snadné přizpůsobení - upravování stylů se provádí vytvořením nových CSS pravidel, která se vykonávají místo standardních. Nemusí se při tom používat atributy typu "!important"
- Velké množství šablon - šablony v Bootstrapu umožňují měnit podle potřeb již změněné elementy. Mnoho vývojářů nabízí své vlastní šablony (placené i zdarma).
- Obrovská komunita vývojářů
- Široká škála použití - Bootstrap se používá při tvorbě témat pro téměř všechny CMS (OpenCart Prestashop, Magento, Joomla, Bitrix, WordPress aj.) včetně jednostránkových aplikací [7]

**Základní nástroje bootstrapu:**

- Sítě - předem definované rozměry sloupečků, které mohou být okamžitě použity, například šířka sloupečku 90px odpovídá třídě .span2, kterou lze použít v popisu CSS dokumentu
- Šablony - pevná nebo pohyblivá šablona dokumentu
- Typografie - popis písma, definice některých tříd pro písma jako je kód, citáty atd.
- Média - umožňuje kontrolu obrázků a videí
- Tabulky - nástroje formátování tabulek a přidání funkcí např. pro možnost třídění
- Formuláře - třídy pro formátování formulářů a některé funkce s nimi
- Navigace - třídy pro formátování záložek, stránek, menu a panelů nástrojů
- Upozornění - formátování dialogových oken, nápovědy a pop-up oken. [7]

### 2.1.6 JavaScript

JavaScript je skriptovací jazyk, pomocí kterého je možné vytvářet interaktivní HTML dokumenty, provádět výpočty a ověřování dat bez přístupu k serveru. Je to nejpopulárnější skriptovací jazyk na internetu a funguje ve většině prohlížečů jako je Internet Explorer, Firefox, Chrome, Opera a Safari. K realizaci skriptu nejsou potřeba žádné doplňující programy. Skript je interpretován a zpracován interpretem zabudovaným v prohlížeči. Prohlížeč začne okamžitě realizovat kód JavaScriptu, jakmile ho najde na stránce. Každý řádek kódu se realizuje postupně, přechod na další řádek se provede pouze po vykonání předchozího. Pokud dokument obsahuje více skriptů, budou se realizovat v pořadí, v jakém se objevují v dokumentu. [8]

#### 2.1.6.1 Připojení skriptů k HTML dokumentu

Skripty JavaScript jsou buď interní, tj. jejich obsah je součástí HTML dokumentu, nebo externí, uložené v samostatném souboru s příponou js. Skripty je možné začlenit do HTML dokumentu následujícími způsoby:

- **ve formě hypertextového odkazu** - je nutné umístit kód do samostatného souboru a připojit odkaz na soubor v záhlaví nebo tělu stránky. Tento způsob se obvykle používá u rozsáhlých skriptů nebo skriptů opakovaně používaných na různých webových stránkách.

- **ve formě obsluhy události** - každý HTML element má události, které se spouštějí v určitém okamžiku. Potřebná událost se do HTML elementu přidává pomocí atributu. Jako hodnotu tohoto atributu je nutné zadat název funkce pro obsluhu události, která se spustí při jejím vzniku. V důsledku spuštění události se provede odpovídající kód obsažený ve funkci. Tento způsob se používá především pro krátké skripty. Například takto můžete nastavit změnu barvy pozadí při stisknutí tlačítka apod.
- **Uvnitř elementu `<script>`** - Element `<script>` může být vložen na libovolné místo v HTML dokumentu. Obvykle je kód JavaScript umístěn v záhlaví dokumentu (element `<head>`), nebo uvnitř tagu `<body>`. Uvnitř tagu `<script>` je umístěn kód, který se provádí bezprostředně po přečtení prohlížečem, nebo obsahuje popis funkce, která se provede po její zavolání (obsluha události). Kód funkce může být umístěn kdekoliv uvnitř tagu `<script>`, důležité je, aby byl načten v momentu zavolání funkce. [9]

#### 2.1.6.2 Knihovna jquery

jQuery je volně šiřitelná knihovna jazyka JavaScript, která obsahuje připravené funkce JavaScriptu. Všechny operace jQuery jsou prováděny z kódu JavaScriptu. Knihovna jQuery usnadňuje manipulaci s HTML elementy, řízení jejich chování a používá Document Object Model (DOM) ke změně struktury webové stránky. Elementy se vybírají pomocí selektorů CSS. Výběr se provádí pomocí konstrukce `$("selektor")`. Tato konstrukce vrací objekt JQuery, který obsahuje žádný nebo více elementů DOM a umožňuje interakci s nimi. [10]

**Možnosti jQuery:**

- přístup k libovolné položce DOM a manipulace s ní
- práce s událostmi
- snadná implementace různých vizuálních efektů
- práce s AJAX - podpora asynchronního zpracování webových stránek
- obsahuje velké množství pluginů JavaScriptu pro tvorbu elementů uživatelských rozhraní [8]

## 2.2 Technologie pro tvorbu serverové části

### 2.2.1 Platforma Java EE

Pro tvorbu serverové části aplikace je použita platforma Java Enterprise Edition (Java EE). Základem Java EE je Java Standard Edition (Java SE). Java EE však navíc poskytuje knihovny, které usnadňují vývoj komplexních enterprise (podnikových) internetových aplikací.

Termín Java označuje kromě programovacího jazyka také platformu. Java platforma se skládá z virtuálního stroje a příslušného API (Application Programming Interface). Virtuální stroj je prostředí, ve kterém běží aplikace, respektive třídy přeložené do byte kódu. Virtuální stroj se distribuuje jeho výrobcem pro více hardware a operačních systémů (například 32 nebo 64 bitová architektura, Windows, Linux apod.). API je sada předprogramovaných tříd, které jsou k dispozici při psaní vlastních tříd a umožňují přístup k funkcionalitě virtuálního stroje. Právě v obsahu API se od sebe jednotlivé Java platformy liší. [11]

Existují tři Java platformy: Java SE, Java ME a Java EE. Java EE je nadstavbou nad Java SE. To znamená, že pokud chceme na nějakém počítači provozovat Java EE aplikaci, musí na něm být nainstalováno také standardní API, které je enterprise aplikacemi využíváno. Java EE poskytuje okolo

dvou desítek API tříd, z nichž každá usnadňuje vývoj v určité oblasti. Jako příklad můžeme uvést Java Servlet API nebo JavaServer Pages (JSP), které poskytují podporu dynamických webových stránek. [11]

### 2.2.2 Aplikační server

Serverovou aplikaci zastřešující všechny knihovny, které dle specifikací Java EE platformy zajišťují požadovanou funkcionalitu, označují pojmem aplikační server. Aplikační server tvoří vrstvu mezi operačním systémem a enterprise aplikacemi. Enterprise aplikace je aplikací, na kterou jsou kladeny vyšší nároky co se týče spolehlivosti, dostupnosti, robustnosti a výkonnosti, například potřeba obsloužit současně velké množství požadavků (od klientů). Typickými zástupci enterprise aplikací jsou moderní webové aplikace. Požadavky od klientů zpracovávají aplikační servery, které mohou pracovat v režimu iterativním, kdy vyřizují požadavky postupně, nebo v režimu konkurentním, kdy přijímají a zpracovávají souběžně více požadavků najednou [12]. Podobně, jako operační systém poskytuje základní funkce programům (například pro přístup k souborovému systému), poskytuje aplikační server různé funkce enterprise aplikacím. Příkladem takových funkcí mohou třeba být podpora transakčního zpracování požadavků, výměna zpráv mezi různými aplikacemi a další. Kromě toho poskytuje aplikační server další typické služby jako např. administrátorskou konzoli, logování apod. [13]

Většina aplikačních serverů je vyvíjena v programovacím jazyce Java. Je to dané nejenom tím, že je Java cross-platformní, ale také tím, že v Java existuje standard pro implementaci enterprise aplikací - Java Enterprise Edition (JEE). [13]

Na aplikační server se nasazují tzv. komponenty, což jsou základní jednotky, ze kterých je složena výsledná aplikace. Komponent existuje několik druhů, z nichž jsou pro tuto práci použité pouze webové komponenty: servlety, JSP soubory a JSF soubory. Ze známých aplikačních serverů lze zmínit například JBoss od firmy Red Hat či GlassFish od firmy Sun Microsystems. Ovšem nejvíce používaným serverem pro menší aplikace je aplikační server Apache Tomcat, který je použitý v této práci. Apache Tomcat je volně šiřitelný projekt, který umožňuje fungování jak statických souborů, tak i dynamické části webové stránky. Realizuje specifikaci kontejneru servletů a specifikaci JavaServer Pages (JSP). Je použit jako samostatný server webové aplikace. [13]

### 2.2.3 Eclipse

Pro návrh návrh a ladění enterprise aplikací lze použít širokou škálu prostředků, počínaje jednoduchými textovými editory až po vysoce sofistikovanými vývojovými prostředí (IDE). Ale právě IDE poskytuje vývojáři nejvíce možností, usnadňuje jeho práci a zkracuje čas potřebný pro vývoj aplikace. Lze vyjmenovat několik často používaných IDE, jako například: NetBeans, IntelliJ IDEA nebo Eclipse.

Eclipse je volně šiřitelná softwarová platforma s otevřeným zdrojovým kódem, kontrolovaným organizací Eclipse Foundation. Je napsaná v programovacím jazyce Java a hlavním smyslem jejího vzniku je zvýšit produktivitu procesů vývoje softwaru, jelikož umožňuje automatizovat vytvoření a ladění programu. Eclipse používá knihovnu závislou na platformě SWT - Standard Widget Toolkit. [14]

**Hlavní vlastnosti platformy Eclipse:**

- je cross-platformní - pracuje pod operačními systémy Windows, Linux, Solaris a Mac OS X
- umožňuje programování v různých jazycích, jako je Java, C a C++, PHP, Perl, Python a další

- je rámcem pro vývoj dalších nástrojů a nabízí rozsáhlou sadu API pro vytváření modulů nástroj pro tvorbu téměř libovolného klientského software díky použití přístupu RCP (Rich Client Platform) [14]

## 2.3 FITLayout Framework

FITLayout Framework je rozšiřitelný rámec pro segmentaci a analýzu webových stránek napsaný v jazyce Java. Definuje Java API pro reprezentování renderované webové stránky a její rozdělení na vizuální oblasti a jejich další analýzu. Také poskytuje základ pro implementaci segmentačních algoritmů stránky se společným aplikačním rozhraním. FITLayout Framework obsahuje různé nástroje pro zpracování výsledků segmentace pomocí různých textových nebo vizuálních klasifikačních metod. [1]

### 2.3.1 Architektura

FITLayout Framework pracuje s renderovanou webovou stránkou reprezentovanou stromem boxů. Strom boxů se získá vykreslením stránky a výpočtem pozic, písma, barev a dalších vizuálních vlastností jednotlivých částí jejího obsahu (boxů). Strom boxů je vstupem pro segmentační algoritmy. [1]

Hlavním úkolem realizovaným ve FITLayout Framework je segmentace stránky. Analyzuje vstupní strom boxů a vytváří strom vizuálních oblastí, které odpovídají zjištěným vizuálním blokům na stránce. Vytvořený strom vizuálních oblastí může být dále zpracován operátory stromu oblastí, které reprezentují nezávislé kroky následného zpracování segmentace. Tyto kroky mohou změnit organizaci výsledného stromu vizuálních oblastí, např. seskupit několik uzlů do nových oblastí atd. [1]

### 2.3.2 Moduly

FITLayout framework se skládá z těchto základních modulů:

- API - základní rozhraní Java a jejich generické implementace, které definují společné aplikační rozhraní pro metody segmentace stránky (viz níže) výchozí implementace vykreslené zdrojový kód stránky založené na CSSBox renderovací systém.
- CSSBox provázání - výchozí implementace zdroje renderované stránky založené na renderovacím mechanismu CSSBox.
- Segmentace - implementace základní metody segmentace stránky, která může být dále rozšířena přidáním vlastních operátorů stromu oblastí.
- Nástroje - nástroje pro řízení procesu segmentace, které obsahují grafický prohlížeč výsledku segmentace. [1]

FITLayout Framework poskytuje následující Java balíčky:

- `org.fit.layout.model` - základní rozhraní Java použité pro reprezentaci vykreslené stránky (strom boxů) a výsledku segmentace (strom oblastí).
- `org.fit.layout.impl` - výchozí implementace rozhraní z balíku modelu. Tyto implementace se mohou použít jako výchozí bod pro další rozšíření.
- `org.fit.layout.api` - rozhraní určené pro samotný FITLayout Framework, zahrnující služby různých druhů.
- `org.fit.layout.gui` - společné rozhraní GUI prohlížeče sloužící k monitorování zpracování stránky. [1]

### 2.3.3 Služby

Architektura FITLayout je snadno rozšiřitelná vytvořením nových pluginů, které poskytují nové funkce, jako například nové zdroje stromu boxů (renderery dokumentu), algoritmy segmentace, operátory následně zpracovávaných stromů oblastí nebo rozšíření GUI.
Rozlišují se následující typy služeb:

- `BoxTreeProvider` - zdroj stromu boxů, tj. rendereru stránky. Na základě vstupních parametrů (např. URL stránky), vykreslí stránku a vytvoří strom boxů.
- `AreaTreeProvider` - zdroj stromu oblastí; tj. základní algoritmus segmentace. Dostane strom boxů na vstupu a vytvoří strom vizuálních oblastí, který reprezentuje segmentovanou stránku.
- `AreaTreeOperator` - následné zpracování operace použité na strom vizuálních oblastí. Může vykonávat různé operace se stromem jako spojování uzlů, rozdělování uzlů, rozšíření hierarchie atd.
- `LogicalTreeProvider` - analyzátor, který dostane konečný strom oblastí na vstup a přiřadí sémantiku do vybraných oblastí (uzly stromu). [1]

### 2.3.4 Box Tree

Celá vykreslená stránka je reprezentovaná pomocí objektu `Page`. Jeho metoda `getRoot()` získává kořenový uzel stromu boxů, který reprezentuje obsah stránky. Uzly stromu boxů jsou tvořeny objekty Box, které představují jednotlivé vykreslené boxy. Každý box má pevnou pozici na vykreslené stránce získanou pomocí metody `getBounds()` a některé další vizuální vlastnosti, jako je velikost písma, barvy atd. [1]
Existují následující typy boxu:

- `ELEMENT` - box generovaný elementem DOM
- `TEXT_CONTENT` - box představující zobrazený text
- `REPLACED_CONTENT` - box představující nahrazený obsah (obrázek nebo jiný objekt) [1]

Boxy jsou organizovány do hierarchické struktury. Pro přecházení hierarchií mohou být použity metody `getParentBox()`, `getChildBox()` a `getChildCount()`. Boxy `TEXT_CONTENT` a `REPLACED_CONTENT` jsou vždy koncové uzly stromu. Uzly `ELEMENT` se mohou vyskytovat kdekoliv ve stromě. [1]

#### 2.3.4.1 Zdroj boxu CSSBox

Výchozí zdroj boxu je implementován v modulu `layout-cssbox` (`CSSBox bindings`) jako třída `CSSBoxTreeProvider`. Jednotlivé boxy jsou reprezentovány pomocí objektů `BoxNode`. Zdroj boxu CSSBox vykresluje vstupní dokument identifikovaný jeho URL adresou. Podporuje HTML/CSS a PDF dokumenty. Je založen na open-source CSS Box renderovacím mechanismu. [1]

### 2.3.5 Segmentace

Algoritmus segmentace bere strom boxů na vstupu a vytváří strom vizuálních oblastí. Výsledný strom je reprezentován objektem `AreaTree`. Jeho metoda `getRoot()` získává kořenový uzel tohoto stromu oblastí, který představuje výsledek segmentace. Každý uzel ve stromu oblastí je reprezentován pomocí objektu Area, který odpovídá vizuální oblasti detekované na stránce. Kořenový uzel odpovídá celé stránce, uzly potomků odpovídají menším zjištěným oblastem. Koncové oblasti mohou obsahovat aktuální boxy ze stromu boxů, které představují obsah oblastí. Uzly poskytují základní metody pro navigaci a manipulaci ve stromě. Všechny tyto funkce jsou specifikovány ve společným

rozhraní `AreaTreeNode`. Pozice oblasti ve vykreslené stránce a všechny její vizuální elementy, jako jsou písma a barvy mohou být získány prostřednictvím interface `ContentRect`. [1]

### 2.3.6 Nástroje

Modul `Tools` poskytuje nástroje pro spouštění a řízení procesu segmentace. Obsahuje třídu `Processor`, která implementuje celý proces segmentace a třídu `BlockBrowser` s grafickým uživatelským rozhraní. [1]

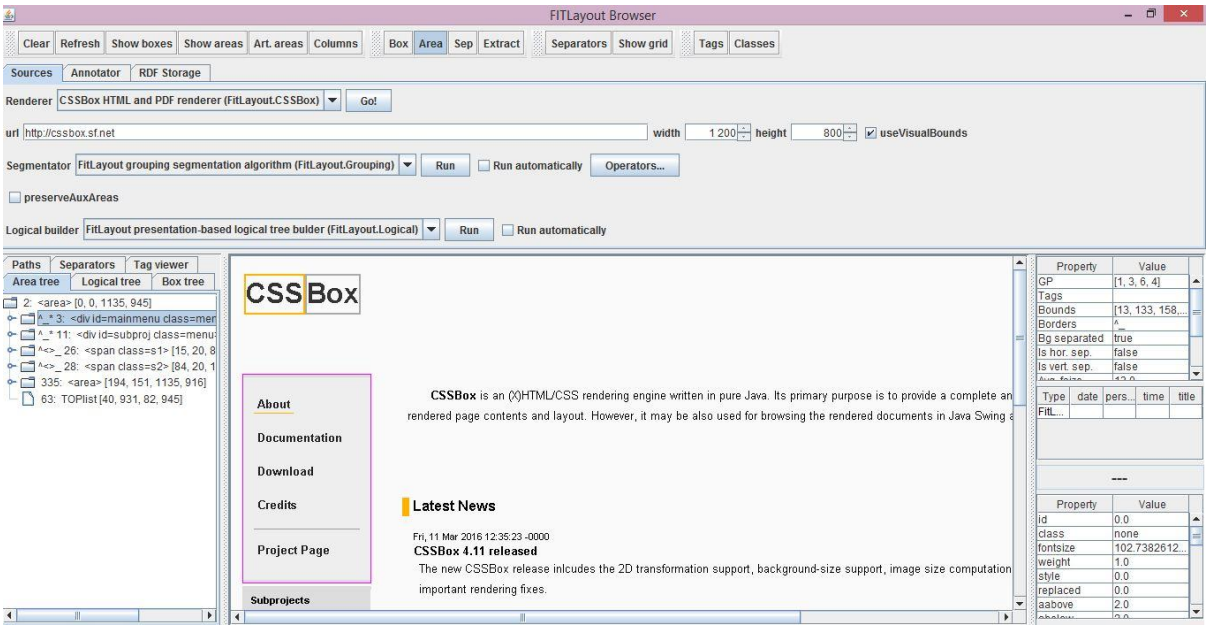
#### 2.3.6.1 Processor

`Processor` je třída, která odpovídá za provedení celého procesu segmentace, tj. vytvoření stromu základních vizuálních oblastí a aplikování definovaných operátorů na tento strom. Základní funkcionalita je definována v abstraktní třídě `BaseProcessor`, která umožňuje dvě implementace:

- `ScriptableProcessor` používající JavaScript pro konfiguraci operátorů oblasti, které budou aplikovány
- `GUIProcessor`, kde může být konfigurace operátorů změněna [1]

# 3. Návrh řešení

Hlavním cílem této práce bylo vytvořit webovou verzi desktopové aplikace FITLayout Framework [1] na základě dynamické webové stránky a JavaScriptu. Grafické uživatelské rozhraní FITLayout Framework je uvedena na obrázku 3.1.




*Obrázek 3.1 Grafické uživatelské rozhraní FITLayout Framework*

Na základě provedených analýz a nastudování teoretických materiálů bylo navrženo řešení aplikace, které je popsáno v níže uvedených kapitolách.

## 3.1 Základní funkční požadavky

### 3.1.1 Seznam realizovaných funkcí

Seznam hlavních funkcí webové aplikace je určen funkcemi desktopové aplikace FITLayout Framework. [1]

Aplikace bude realizovat následující funkce:

1. Zadání adresy www stránky
2. Nastavení šířky renderované stránky
3. Nastavení výšky renderované stránky
4. Výběr rendereru
5. Spuštění rendereru
6. Výběr segmentátoru
7. Spuštění segmentátoru
8. Vizualizace renderované webové stránky
9. Generování stromů oblastí a boxů
10. Přepínání mezi označení oblastí (`Area`) a boxů (`Box`)
11. Označení boxů a oblastí
12. Odstranění označení boxů a oblastí
13. Výběr aktivního stromu

14. Označení položky stromu na stránce
15. Výpis vlastností oblasti
16. Obnovení výchozího stavu stromu
17. Označení boxu vybraného elementu ve stromu boxů a boxů všech jeho potomků
18. Označení oblasti vybraného elementu ve stromu oblastí a oblastí všech jeho potomků

### 3.1.2 Detailní popis funkcí

**1. Zadání adresy www stránky**
URL adresa stránky se zadává do pole `url` ve správném a úplném tvaru dle specifikace HTML. Při zadání v nesprávném tvaru bude zobrazeno upozornění.

**2. Nastavení šířky renderované stránky**
Šířku stránky lze nastavit v poli označeném `width` a to buď hodnotou zadanou z klávesnice nebo pomocí tlačítek v boxu. Lze zadávat pouze nezáporné číselné hodnoty. Výchozí hodnota je 1200 pixelů. Minimální hodnota je 768 pixelů.

**3. Nastavení výšky renderované stránky**
Výšku stránky lze nastavit v poli označeném `height` a to buď hodnotou zadanou z klávesnice nebo pomocí tlačítek v boxu. Lze zadávat pouze nezáporné číselné hodnoty. Výchozí hodnota je 800 pixelů. Minimální hodnota je 360 pixelů.

**4. Výběr rendereru**
Renderer se vybírá v rozbalovacím boxu s názvem `Renderer` pomocí myši nebo tlačítek.

**5. Spuštění rendereru**
Vybraný renderer se spouští tlačítkem `Go!`. Po zpracování vybrané stránky se zobrazí obsah renderované stránky v prostředním okně a strom oblastí stránky se zobrazí v záložce `Box Tree`.

**6. Výběr segmentátoru**
Segmentátor se vybírá v rozbalovacím boxu s názvem `Segmentator` pomocí myši nebo tlačítek.

**7. Spuštění segmentátoru**
Vybraný segmentátor se spouští tlačítkem `Run` nebo zaškrtnutím boxu `Run Automatically` a zmáčknutím tlačítka `Go!`.

**8. Vizualizace renderované webové stránky**
Stránka se zobrazí v prostředním okně spodní části dle zadané výšky a šířky.

**9. Generování stromů oblastí a boxů**
- strom boxů (Box Tree) se vygeneruje po spuštění rendereru nebo segmentátoru
- strom oblastí (Area Tree) se vygeneruje po spuštění segmentátoru

**10. Přepínání mezi označení oblastí (`Area`) a boxů (`Box`)**
Přepínání se provádí pomocí tlačítek `Box` a `Area` v tlačítkové liště s fixací v horní části obrazovky. Tlačítko `Area` umožňuje označení oblastí, tlačítko `Box` umožňuje označení boxů. Nelze zmáčknou obě tlačítka naráz.

**11. Označení boxů a oblastí**

Označení plochy se provede kliknutím levým tlačítkem myši na renderované stránce

- pokud je zmáčknuté tlačítko `Box`, box se barevně vyznačí a zobrazí se jako aktivní ve stromu Box Tree
- pokud je zmáčknuté tlačítko `Area`, oblast se barevně vyznačí a zobrazí se jako aktivní ve stromu Area Tree
- pokud není zmáčknuté ani jedno tlačítko, nelze vyznačovat elementy ani oblasti na renderované stránce

**12. Odstranění označení boxů a oblastí**

Zmáčknutím tlačítka `Clear` se vymažou označené plochy na renderované stránce.

**13. Výběr aktivního stromu**

- pro zobrazení stromu elementů (Box Tree) je nutné kliknout na záložku `Box Tree`
- pro zobrazení stromu oblastí (Area Tree) je nutné kliknout na záložku `Area Tree`

**14. Označení položky stromu na stránce**

- po kliknutí na položku v Box Tree se na stránce barevně označí plocha, která odpovídá souřadnicím elementu v Box Tree
- po kliknutí na položku v Area Tree se na stránce barevně označí plocha, která odpovídá souřadnicím oblasti v Area Tree
- je možné označování elementů resp. oblastí pomocí kliknutí myši na plochu na stránce dle aktuálního nastavení tlačítek `Box` a `Area`

**15. Výpis vlastností oblasti**

Vlastnosti se vypisují do tabulky, kde levý sloupec (`Property`) obsahuje název a pravý sloupec (`Value`) obsahuje odpovídající hodnotu. Vlastnosti jsou vypsány pro aktuálně označenou oblast v Area Tree.

**16. Obnovení výchozího stavu stromu**

Obnovení výchozího stavu stromu se provede kliknutím na tlačítko `Refresh`. Při tom se u stromů rozbalí kořenový uzel a všechny ostatní se sbalí.

**17. Označení boxu vybraného elementu ve stromu boxů a boxů všech jeho potomků**

Tato funkce se provede kliknutím na tlačítko `Show boxes`. Přitom bude označen na stránce box vybraného elementu ve stromu boxů a také boxy všech jeho potomků.

**18. Označení oblasti vybraného elementu ve stromu oblastí a oblastí všech jeho potomků**

Tato funkce se provede kliknutím na tlačítko `Show areas`. Přitom bude označena na stránce oblast vybraného elementu ve stromu oblastí a také oblasti všech jeho potomků.

### 3.1.3 Požadavky na vstupní data

Vstupními daty pro aplikaci jsou:

1. url adresa analyzované webové stránky
2. rozměry oblasti, do které se renderuje webová stránka

3. vybraný renderer a segmentátor

Vstupní data se zadávají uživatelem ručně do odpovídajících polí webové aplikace.

### 3.1.4 Požadavky na výstupní data

Výstupem aplikace je vizuální výstup, který je zobrazení renderované stránky ve zvolené oblasti a také výpis vlastností označené oblasti.

### 3.1.5 Časové požadavky

Časové požadavky k běhu aplikace nejsou stanoveny.

## 3.2 Spolehlivost fungování aplikace

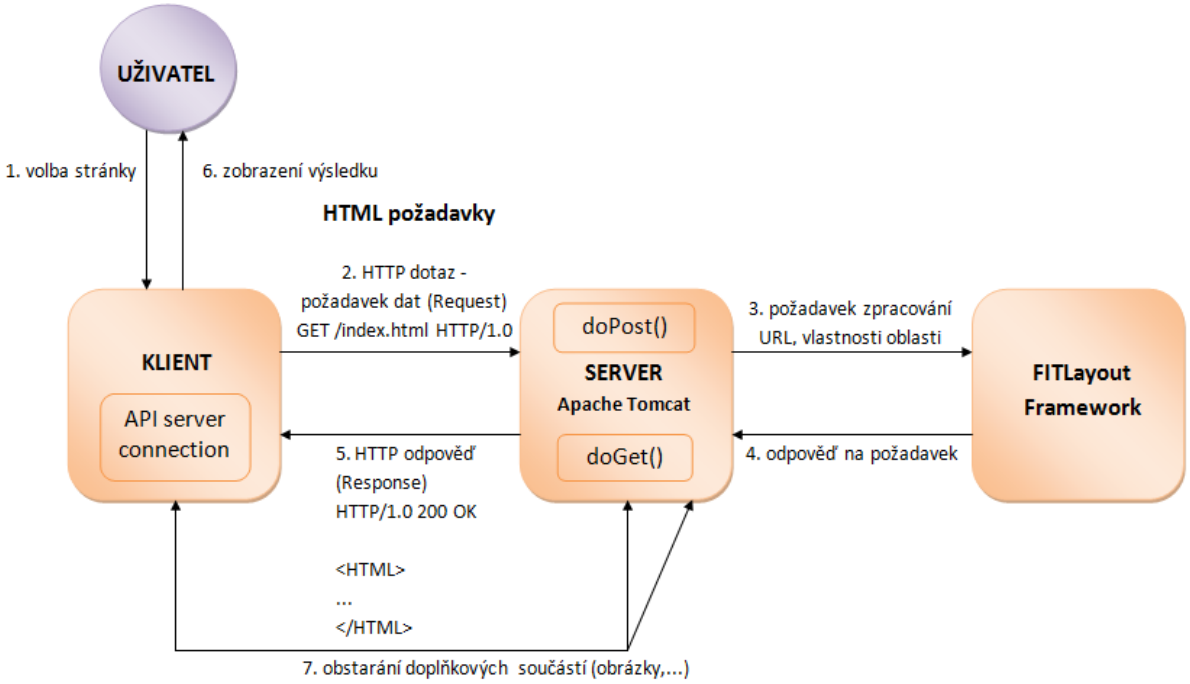
Při realizaci aplikace byly použity ověřené softwarové technologie a postupy programování (Java, JavaScript, HTML, CSS), které zajišťují spolehlivou práci aplikace nezávisle na použitém uživatelem operačním systému a hardware. V aplikaci jsou použita opatření zabraňující uživateli zadávání nekorektních vstupních dat, která by v důsledku mohly vést k nesprávné funkci resp. havárii aplikace.

## 3.3 Požadavky na kvalifikaci uživatele

Navržená aplikace nepožaduje od uživatele speciálních znalostí v oblasti informačních technologií a může tuto aplikaci ovládat každý, kdo je schopen práce s internetovým prohlížečem a je seznámen s fungováním běžných operačních systémů (spuštění a ukončení programu) a je obeznámen s popisem funkcí aplikace.

## 3.4 IT požadavky k aplikaci

Pro návrh aplikace je použita architektura klient-server s web klientem. Jako klientský software může být použit jakýkoliv běžně rozšířený HTML prohlížeč podporující jazyk HTML5. Základní architektura aplikace je uvedena na obrázku 3.2.




*Obrázek 3.2 Základní architektura aplikace*

Pro tvorbu zdrojových kódů jsou použity cross-platformní programovací jazyky Java a JavaScript.

Funkcionalita aplikace je rozdělena mezi aplikačním serverem a klientskou částí následujícím způsobem:

- serverová část:
  - přijímá požadavky a vstupní data od klienta
  - načítá požadovanou stránku
  - provádí analýzu segmentace webových stránek
  - generuje stromy oblastí a elementů
  - generuje renderovanou stránku
  - načítá informace o vlastnostech vybrané oblasti
  - předává klientské části požadované informace
- klientská část zajišťuje:
  - zadání a kontrolu vstupních dat
  - vizualizaci
  - označení vybrané oblasti
  - přepínání mezi označení oblastí a elementů

Pro realizaci serverové části bude použitý programovací jazyk Java verze JRE 1.7. Statické prvky webové stránky jsou realizovány pomocí jazyků HTML5, CSS 2.x a CSS knihovny Bootstrap 3.3.5. Dynamická část stránky je realizovaná pomocí JavaScript a knihoven jQuery 1.12.0 a Bootstrap 3.3.5.

### 3.4.1 Programové prostředky použité v aplikaci

Pro práci aplikace jsou požadovány následující programové prostředky:
1. aplikační server:

- Apache Tomcat 7.0 nebo vyšší
- Java Runtime Environment 1.7 nebo vyšší
- FITLayout Framework

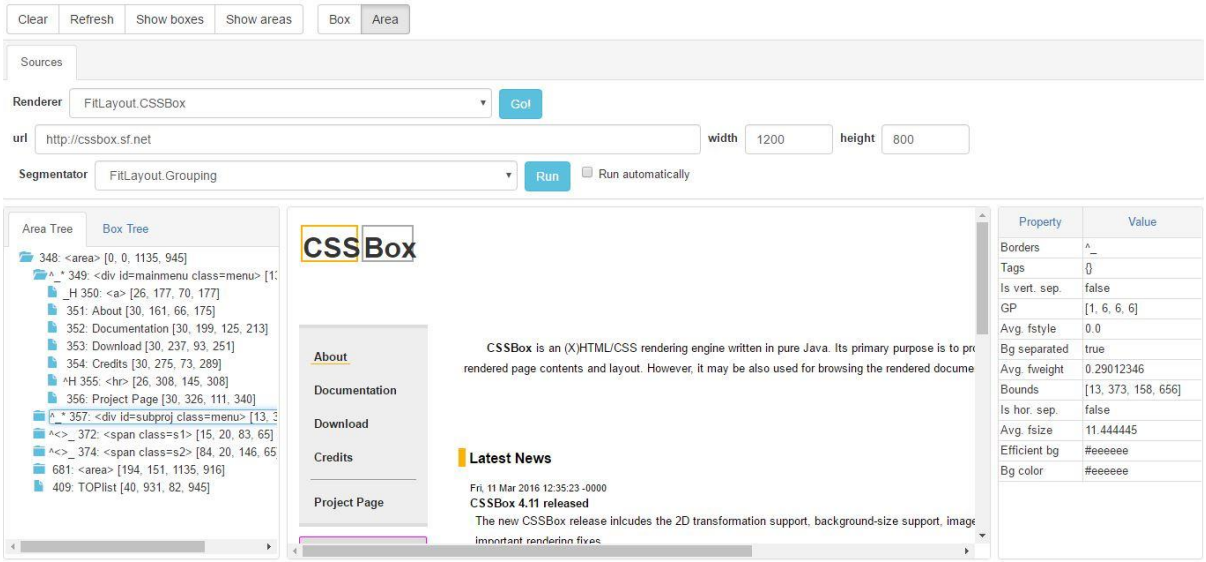
2. klientské programové vybavení:

- běžně dostupný HTML prohlížeč s podporou HTML5 (Chrome, Firefox, IE aj.). Program se navrhoval v prostředí operačního systému Windows® 8.1.

Pro návrh a ladění aplikace bylo použito prostředí Eclipse Java EE IDE for Web Developers, ver. Kepler Service Release 2.

## 3.5 Uživatelské rozhraní

Uživatel komunikuje s aplikací prostřednictvím grafického uživatelského rozhraní. Návrh uživatelského rozhraní vychází z funkcí aplikace a z uživatelského rozhraní FITLayout Framework a je uveden na obrázku 3.3.




*Obrázek 3.3 Uživatelské rozhraní dynamické webové stránky*

Jazyk uživatelského rozhraní - anglický. Uživatelská oblast bude rozdělena na několik funkčních částí. V horní části budou umístěné ovládací prvky (tlačítka) seskupené do tlačítkových lišt, pomocí kterých uživatel může ovládat vizualizaci výsledku analýzy webové stránky.
V prostřední části budou seskupeny ovládací prvky, které slouží k zadání vstupních dat:

- pole pro zadání url
- pole pro nastavení výšky a šířky renderované stránky
- rozbalovací boxy pro výběr rendereru a segmentátoru
- výběrový box pro automatické spouštění segmentátoru
- tlačítka pro spuštění renderování stránky a její segmentace

Spodní část, která ale má největší rozměry, slouží ke zobrazení výsledků analýz:

- vlevo se nachází panel se záložkami pro zobrazení stromu oblastí nebo boxů
- uprostřed se nachází oblast pro zobrazení renderované stránky
- vpravo se nachází tabulka pro zobrazení vlastností aktuálně vybrané oblasti

## 3.6 Spouštění a ukončení aplikace

Aplikace se bude spouštět zadáním názvu stránky ve webovém prohlížeči. Aplikace nevyžaduje žádnou autorizaci (uživatelské jméno, heslo). Standardním způsobem ukončení aplikace je zavření okna prohlížeče nebo ukončení práce prohlížeče.

# 4. Implementace

Implementace aplikace vychází z návrhu řešení a rozděluje se na implementaci serverové části a klientské části (implementace uživatelského rozhraní a implementace chování aplikace - JavaScript). Veškeré informace, které budou následně zobrazeny na webové stránce získává serverová část z balíků FITLayout Framework. Tyto informace sebou představují datové struktury platformy Java. Proto při implementaci bylo třeba nastudovat formát datových struktur v Javě, navrhnout způsob jejich vizualizace pomocí prostředků HTML a CSS a následně napsat kód, který převádí datovou strukturu Java na běžný text podle pravidel jazyka HTML. Základními HTML elementy, které se budou zobrazovat je rozbalovací strom, kterému odpovídají objekty Java třídy `Box` a `Area`, tabulka vlastností oblasti a obsah renderované stránky. Rozbalovací strom musí navíc realizovat funkce stromu Java Swing: rozbalení uzlu, sbalení uzlu, označení jednotlivých položek. Tyto funkce budou realizované pomocí JavaScript v klientské části. Detailní popis řešení zmíněných problémů a implementace uvedených elementů je popsán v následujících kapitolách.

## 4.1 Implementace uživatelského rozhraní

K implementaci uživatelského rozhraní byly použity jazyk HTML5, CSS2 a také knihovna CSS Bootstrap 3.3.5. Uživatelské rozhraní je realizované pomocí dynamické HTML stránky. Její základní parametry jsou nastavené v hlavičce `head`.

První meta tag oznamuje prohlížeči typ dokumentu text/html a jeho kódování UTF-8.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Druhý meta tag říká prohlížeči aby nastavil `viewport` (pohled) na celou šířku zařízení a zachoval původní měřítko.

```
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

### 4.1.1 Tlačítková lišta

Tlačítka jsou seskupená do dvou tlačítkových lišt:


*Obrázek 4.1 Tlačítková lišta*

**1) Kontrola vizualizace stránky**

Tlačítková lišta je realizovaná s použitím třídy `btn-group` knihovny Bootstrap.

```
<div class="btn-group">
```

Samotná tlačítka jsou realizována pomocí třídy `btn btn-default` (bez pozadí).

```
<button id="btn-clear" type="button" class="btn btn-default">
```

Tlačítková lišta obsahuje následující tlačítka:

- tlačítko `Clear` (`id="btn-clear"`) - odstraňuje barevné ohraničení boxů a oblastí
- tlačítko `Refresh` (`id="btn-refresh"`)- uvádí stromy boxu a oblastí do výchozího stavu
- tlačítko `Show Boxes` (`id="btn-boxes"`) - označuje na stránce box vybraného elementu ve stromu boxu a také boxy všech jeho potomků
- tlačítko `Show Areas` (`id="btn-areas"`) - označuje na stránce oblast vybraného elementu ve stromu oblastí a také oblasti všech jeho potomků

**2) Výběr zobrazovacích prvků**

Tlačítková lišta je realizovaná s použitím třídy `btn-group` knihovny Bootstrap. V této tlačítkové liště se tlačítka chovají jako přepínače (radio button) tj. při zmáčknutí na tlačítko, se toto tlačítko stane aktivní a ostatní tlačítka v tlačítkové liště neaktivní. Ve výchozím stavu je zmáčknutým (aktivním) tlačítkem tlačítko Area.

```
<button type="button" class="btn btn-default active" id="area">
```

Třída `active` ukazuje prohlížeči, že toto tlačítko je zmáčknuté.

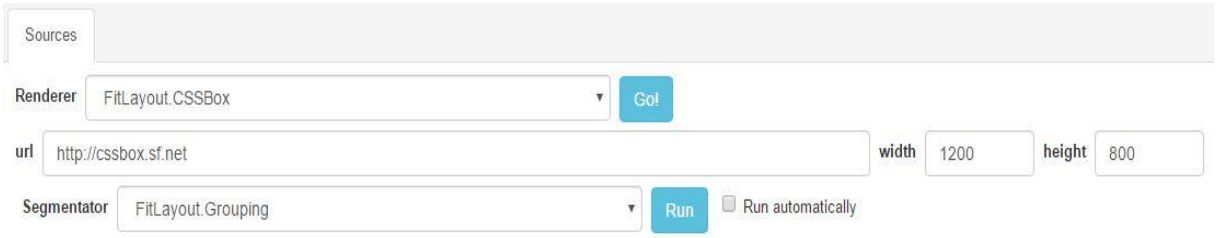
Lišta obsahuje následující tlačítka:

- tlačítko `Box` (`id="btn-box"`) - umožňuje označit boxy na stránce pomocí kliknutí myši
- tlačítko `Area` (`id="btn-area"`) - umožňuje označit oblasti na stránce pomocí kliknutí myši

### 4.1.2 Navigační panel

Navigační panel je realizován pomocí třídy panel panel-default knihovny Bootstrap.

```
<div class="panel panel-default">
```




*Obrázek 4.2 Navigační panel*

#### 4.1.2.1 Sources

V panelu `Sources` jsou umístěny ovládací prvky aplikace, které slouží pro zadání vstupních dat. Ovládací prvky jsou seskupeny ve formě:

```
<form id="form" class="form-inline" action="RendererServlet"
method="post">
```

`action="RendererServlet"` - název servletu, který zpracovává data z této formy
`method="post"` - ukazuje, že se data na server a ze serveru přenáší pomocí požadavku POST

Výběr rendereru a segmentátoru se provádí pomocí elementu `select`.

```
<select class="form-control" name="segmentator">
    <option>FitLayout.Grouping</option>
</select>
```

Element `option` obsahuje jednotlivé položky výběrového boxu.

Pro zadání url adresy se používá nepárový tag input.

```
<input type="url" class="form-control" name="url" size="100"
value="http://cssbox.sf.net">
```

`type="url"` - prohlížeč provede před odesláním základní validaci zadané adresy a eventuálně zobrazí chybové hlášení
`value="http://cssbox.sf.net"` - defaultní hodnota url adresy

Pro zadání šířky a výšky renderované stránky se používá nepárový tag `input`.

```
<input id="width" type="number" class="form-control" name="width"
min="768" value="1200">
<input id="height" type="number" class="form-control" name="height"
min="360" value="800">
```

Atribut `type="number"` - ukazuje, že pole slouží pro zadávání číselných hodnot a zobrazuje se šipkami nahoru a dolů. Atribut `min` nastavuje minimální hodnotu šířky a výšky v pixelech, `value` výchozí hodnotu. Prohlížeč provede po zadání dat základní validaci jejich hodnot a eventuálně zobrazí chybové hlášení.

Pro odeslání formy na server slouží dvě tlačítka (tag `button`) typu submit: `Go!` a `Run`.

```
<button type="submit" class="btn btn-info" name="btn"
value="renderer">
```

Zatrhávací políčko `Run automatically` slouží pro spouštění segmentátoru (resp. builderu) při spouštění rendereru tlačítkem "`Go!`.

```
<input type="checkbox" name="runSegmentator">Run automatically
```

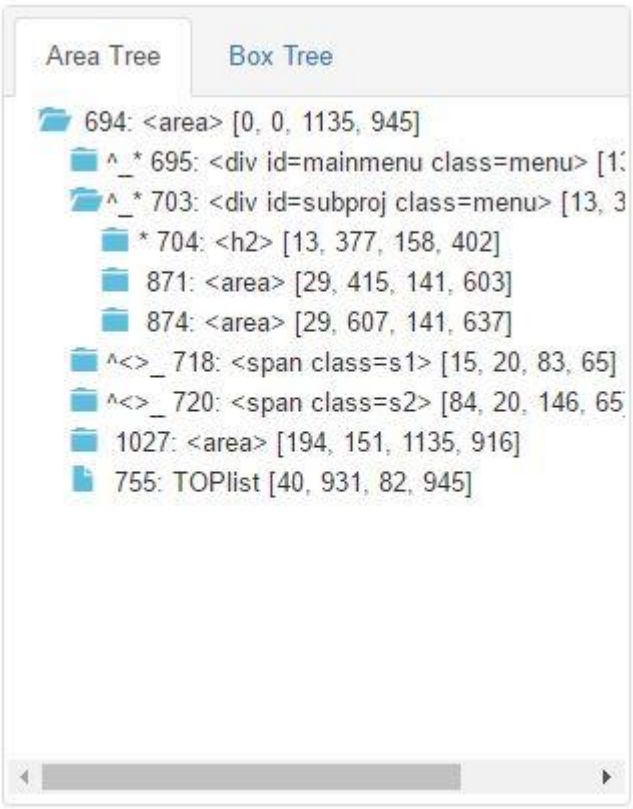
Atribut `name` všech tagů ve formě se používá pro jejich identifikaci a načtení hodnot na serverové straně.

### 4.1.3 Výsledky zpracování webové stránky

Výsledky zpracování se zobrazují pod navigačním panelem se vstupními daty.

#### 4.1.3.1 Navigační panel stromů

Navigační panel stromů je realizován pomocí třídy `panel panel-default` a má následující záložky: `Area Tree` a `Box Tree`. Výchozí otevřená záložka je `Box Tree`.




*Obrázek 4.3 Navigační panel stromů*

#### 4.1.3.2 Realizace stromů

Stromy (Box Tree a Area Tree) jsou realizovány pomocí třídy `tree` a párových tagů `<li> </li>` a `<ul> </ul>` viz níže.

```
<div id="box" class="tree">
    <li class="node">
        <span id="b0">0: Viewport 1200x957...</span>
        <ul>
            <li>
                <span id="b1">+1: div [mainmenu...</span>
            </li>
        ...
        </ul>
        <li>
            <span id="b4">+4: a [] [] B[26, 177, 70, 177]...</span>
        </li>
        ...
    </li>
</div>
```

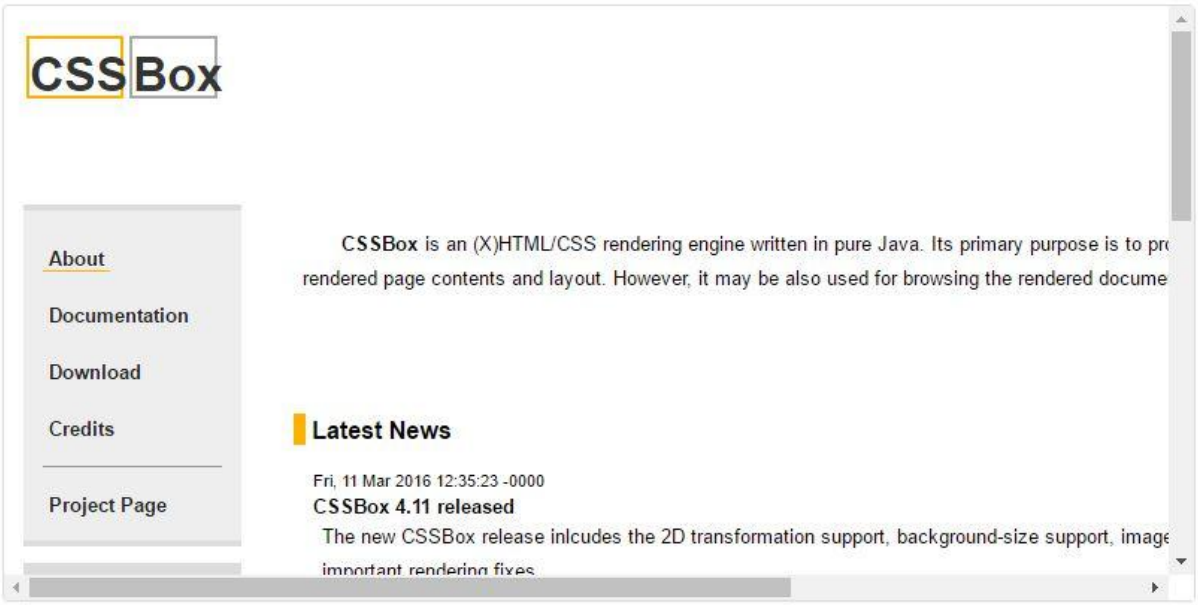
`id="box"` - identifikuje Box Tree, `id="area"` identifikuje Area Tree

Rodiče (uzly) jsou prezentovány párovým tagem `<li> </li>`, který má třídu node.
Potomci se nachází uvnitř párového tagu `<ul></ul>`.
Jednotlivé položky stromu, které nemají potomky jsou prezentovány párovým tagem `<li> </li>`, uvnitř kterého se nachází tag `<span>` obsahující text položky.

#### 4.1.3.3 Obsah stránky

Obsah renderované stránky se získává v serverové části ve formátu HTML pomocí funkcí FITLayout Framework a zobrazuje se v prostřední části jako potomek elementu `div` třídy `panel panel-default`.

```
<div id="page" class="panel panel-default"> </div>
```




*Obrázek 4.4 Obsah stránky*

#### 4.1.3.4 Zobrazení vlastností oblastí

Vlastnosti oblasti, která je vybrána ve stromu Area Tree se zobrazují v tabulce.

| Property | Value |
|---|---|
| Borders | |
| Tags | {} |
| Is vert. sep. | false |
| GP | [0, 1, 2, 1] |
| Avg. fstyle | 0.0 |
| Bg separated | true |
| Avg. fweight | 1.0 |
| Bounds | [13, 377, 158, 402] |
| Is hor. sep. | false |
| Avg. fsize | 11.0 |
| Efficient bg | #dddddd |
| Bg color | #dddddd |

*Obrázek 4.5 Vlastnosti oblastí*

HTML kód tabulky vypadá následovně:

```
<table class="table table-borderer">
    <thead>
        <tr>
            <th>Property</th>
            <th>Value</th>
        </tr>
    </thead>
    <tbody id="info">
         <tr>
            <td>Borders</td>
            <td></td>
        </tr>
        ...
    </tbody>
```

Tabulka je realizovaná pomocí tagu `table`, který má třídu `table table-borderer` a je tvořena dvěma sloupci. Řádky tabulky jsou vytvořeny pomocí párových tagů `<tr> </tr>`. Záhlaví tabulky je prezentováno párovým tagem `<th> </th>`. První sloupec má název `Property` a druhý Value. Ve sloupci `Property` jsou uvedeny názvy vlastností a ve sloupci `Value` jejich hodnoty. Jednotlivé buňky jsou tvořeny tagy `<td> </td>`.

## 4.2 Skripty

Funkcionalitu webové stránky zajišťuje skript napsaný s použitím knihovny jquery a umístěný v souboru `jquery.fitlayout.js`. Skript se připojuje ke stránce pomocí kódu:

```
<script src="jquery.fitlayout.js" type="text/javascript"></script>
```

Skript se spouští po načtení celé stránky. Skript se skládá z podprogramů, které buď obsluhují jednotlivé události (např. zmáčknutí tlačítka) nebo slouží pro vizualizaci oblastí resp. boxů.

### 4.2.1 Odeslání vstupních dat ke zpracování

Data se ke zpracování na server odesílají pomocí funkce,

```
$("#form").submit(function(event)
```

která obsluhuje událost `submit` formy se vstupními daty (`<form id="form">`).

Data se odesílají pomocí požadavku `post`,

```
$.post("RendererServlet", $("#form").serialize()+"&btn="+$btn)
```

kde `RendererServlet` jméno servletu, který zpracovává požadavek, `$("#form").serialize()` je funkce, která vrací řetězec se jmény elementů formy a jejich hodnotami. Do jména `&btn` je zapsáno jméno zmáčknutého tlačítka.

Data ze serveru se zpracovávají ve funkci `.done(function(data)`, kde proměnná `data` obsahuje kód ve formátu HTML, který se přidává na stránku.

### 4.2.2 Obnovení stromů do výchozího stavu

Obnovení stromů do výchozího stavu se provádí pomocí funkcí `refreshTree()`. Prvnímu (kořenovému) elementu `<li>` se nastavuje atribut `class="node"` a bude zobrazen jeho první potomek (element `<ul>`). Ostatní elementy `<ul>` budou skryté a jejich rodičům bude nastaven atribut `class="node collapsed"`.

### 4.2.3 Rozbalení/sbalení uzlů stromu

Po kliknutí na uzel stromu (ikonu složky) se sbalený uzel rozbalí a zobrazí se jeho přímí potomci, rozbalený uzel se sbalí a jeho potomci budou skrytí.

### 4.2.4 Označení položky stromu

Po kliknutí na položku stromu ji bude nastaven atribut `class="selected"` a bude vizuálně označena s použitím stylů CSS. Dále bude označena příslušná oblast nebo box na stránce a pokud byla označena položka stromu oblastí, na server se pošle požadavek post se souřadnicemi levého horního rohu oblasti. Po zpracování serverem budou v tabulce vlastností zobrazeny vlastnosti tohoto elementu.

### 4.2.5 Vizualizace boxů/oblastí na stránce

Vizualizace boxů/oblastí na stránce se provádí použitím funkce `showBounds($item)`, kde `$item` je element stromu, který je třeba vizualizovat. Vizualizace se provede tak, že se na stránku přidá transparentní element `<div>` přes vybranou oblast ve stromě. Pokud takový `<div>` již existuje, nebude provedena žádná akce. Zvýraznění oblasti (barva a styl rámečku) se provede na základě nastavení CSS v CSS souboru.

### 4.2.6 Získání hranic oblastí/boxů

Hranice oblastí/boxů se získají pomocí funkce `getBounds($item)`. Pro vybraný element stromu (`$item`) se z textu načtou jeho hranice - parametry `left`, `top`, `right`, `bottom`, oddělené čárkou. Funkce vrací tyto hodnoty jako pole ze čtyř elementů.

### 4.2.7 Označení oblasti na stránku

Označení oblasti na stránce se provede pomocí funkce zpracování události kliknutí na stránku. Nejdříve se zjistí atribut id elementu, na který bylo kliknuto. Poté se najde element se stejným id ve stromu boxu nebo oblastí dle zmáčknutého tlačítka `Box` nebo `Area`. Tento element bude označen v odpovídajícím stromu, následně budou načteny jeho souřadnice a tato oblast bude zobrazena na stránce.

### 4.2.8 Odstranění označení elementů a oblastí

Odstranění se provede pomocí funkce zpracování události kliknutí na tlačítko `Clear`. Ze stránky budou odstraněny všechny `<div>`, které mají atribut `class="Box"` nebo `"Area"`.

### 4.2.9 Obnovení výchozího stavu stromu

Obnovení výchozího stavu stromu se provede pomocí funkce zpracování události kliknutí na tlačítko `Refresh`. Při tom se u stromů odstraní parametr `class="selected"` z označených elementů, rozbalí se kořenový uzel a všechny ostatní se sbalí.

### 4.2.10 Označení boxu vybraného elementu ve stromu boxů a boxů všech jeho potomků

Tato funkce je realizována pomocí zpracování události kliknutí na tlačítko `Show boxes`. Přitom bude označen na stránce box vybraného elementu ve stromu boxů a také boxy všech jeho potomků.

### 4.2.11 Označení oblasti vybraného elementu ve stromu oblastí a oblastí všech jeho potomků

Tato funkce je realizována pomocí zpracování události kliknutí na tlačítko `Show areas`. Přitom bude označena na stránce oblast vybraného elementu ve stromu oblastí a také oblasti všech jeho potomků.

### 4.2.12 Přepínání mezi tlačítky Box a Area

Přepínání mezi tlačítky je realizováno pomocí funkce zpracování události kliknutí na tlačítko `Box` nebo `Area`. Přitom se tlačítku, na které bylo kliknuto přidá atribut `class="active"` a druhému tlačítku se tento atribut odebere.

## 4.3 Zpracování dat serverem (servlet)

Zpracování dat serverem se provádí pomocí třídy `RendererServlet` odvozené od třídy `HttpServlet` balíku `javax.servlet` platformy Java EE.

```
@WebServlet("/RendererServlet")
public class RendererServlet extends HttpServlet
```

Tato třída obsahuje dvě chráněné (`private`) globální proměnné typu `GUIProcessor` a `AreaTree` a následující metody:

- `doPost` - zpracovává data od webové stránky a vrací výsledky zpracování
- `getAreaInfo` - vrací vlastnosti vybrané oblasti
- `htmlAreaTree` - vytváří HTML kód stromu oblastí
- `htmlBoxTree` - vytváří HTML kód stromu boxů
- `htmlPageContent` - vrací HTML kód renderované webové stránky na základě stromu oblastí `AreaTree` nebo objektu `Page`
- pomocné metody (`borderString`, `colorString`)

### 4.3.1 Zpracování požadavků od webové stránky

Samotné zpracování požadavku `post` od webové stránky se provádí v přepisované metodě `doPost`.

```
@Override
protected void doPost(HttpServletRequest request,HttpServletResponse
response)
throws ServletException, IOException
```

Tato metoda má dva vstupní parametry: `HttpServletRequest` a `HttpServletResponse` a nevrací žádnou hodnotu. Metoda propouští (`throws`) vyjímky `ServletException` a `IOException`.

Metoda zpracovává následující parametry, které předává webová stránka:

- `btn` - jméno zmáčknutého tlačítka
- `runSegmentator` - stav zaškrtávacího políčka Run actomatically
- `height` - výška renderované stránky
- `width` - šířka renderované stránky
- `url` - URL adresa renderované stránky
- `left`, `top` - pozice levého horního rohu elementu renderované stránky, na který bylo kliknuto myší

Tyto parametry se zapisují do proměnných typu `String` pomocí funkce `request.getParameter`.
Výsledky zpracování se posílají pomocí metody `println` třídy `PrintWriter`. Přičemž pro správné zobrazení českých znaků je třeba nastavit správné kódování odesílaných dat.

```
response.setCharacterEncoding("utf-8");
PrintWriter out = response.getWriter();
```

Servlet dostává dva typy požadavků: požadavek na zpracování URL adresy stránky spolu s doplňujícími informacemi (výška, šířka apod.) tj. informace shromážděné v elementu `<form>` nebo požadavek na zobrazení vlastností vybrané oblasti.

Pokud hodnoty obou parametrů `left` a `top` nejsou prázdné (null), jedná se o dotaz na vlastnosti. V tomto případě servlet na základě těchto souřadnic získá potřebné informace s použitím API FITLayout Framework, poté vytvoří HTML kód vlastností, předá jej stránce a ukončí činnost.

Zpracování požadavku na renderování stránky zadané jejím URL se začíná vytvořením nového objektu třídy `GUIProcessor` FITLayout Framework.

```
proc = new GUIProcessor();
```

Dále se vytváří nový objekt třídy `BoxTreeProvider` pomocí metody `getBoxProviders` třídy `GUIProcessor` na základě vybraného rendereru z výběrového boxu na webové stránce. Poté se objektu `BoxTreeProvider` předávají jako parametry URL, výška a šířka renderované stránky.

```
BoxTreeProvider boxProvider =
proc.getBoxProviders().get(rendererType);
boxProvider.setParam("url", url);
boxProvider.setParam("width", Integer.parseInt(width));
boxProvider.setParam("height", Integer.parseInt(height));
```

Následně získáme renderovanou stránku pomocí metody `getPage` třídy `BoxTreeProvider` FITLayout Framework.

```
Page page = boxProvider.getPage();
```

Pokud stránka nemohla být z jakýchkoli důvodu renderována (metoda `getPage` vrátila hodnotu null), servlet pošle na stránku odpovídající hlášení a ukončí svoji práci. Pokud stránka byla úspěšně renderována pomocí metody `htmlPageContent`, získá se její HTML kód a rovněž se vytvoří HTML kód strom boxů.

```
String html = htmlPageContent(page);
String boxTree = htmlBoxTree(page.getRoot());
```

Pokud bylo u segmentátoru zmáčknuto tlačítko `Run` nebo bylo zaškrtnuto políčko `Run automatically`, navíc se provede segmentace stránky. Vytvoří se objekt typu `AreaTreeProvider` pomocí metody `getAreaProviders` `třídy` `GUIProcessor` na základě vybraného segmentátoru z výběrového boxu na webové stránce a získá se strom oblastí pomocí metody `segmentPage` `třídy` `GUIProcessor`.

```
AreaTreeProvider areaProvider =
proc.getAreaProviders().get(segmentatorType);
atree = proc.segmentPage(areaProvider, null);
```

Následně se získá HTML kód renderované stránky na základě stromu oblastí a HTML kód stromu oblastí. Na závěr všechny vytvořené HTML kódy budou odeslány webové stránce pomocí metody `println` třídy `PrintWriter`.

### 4.3.2 Získání HTML kódu renderované stránky

HTML kód renderované stránky se získá pomocí metody `htmlPageContent`. Tato metoda je implementovaná dvakrát: pro vstupní parametr typu `Page` a pro vstupní parametr typu `AreaTree`. HTML kód stránky se získává pomocí třídy `HTMLOutputOperator` FITLayout Framework. Vytvoří se objekt této třídy a objekt třídy `PrintWriter` a pomocí metody `dumpTo` FITLayout Framework se získá HTML kód, který s následně převede do podoby řetězce.

### 4.3.3 Vytvoření HTML kódu stromu boxů

HTML kód stromu boxů se vytváří rekurzivně pomocí metody `htmlBoxTree`. Vstupním parametrem této metody je objekt typu `Box`, představující uzel stromu boxu. Na začátku se metodě jako parametr předává kořenový uzel a následně se prochází všichni jeho potomci. Pokud jeden z potomků je dalším uzlem, začnou se rekurzivně procházet jeho potomci, dokud neprojdeme všechny uzly stromu a koncové elementy. Struktura stromu je popsaná v podkapitole Realizace stromů.

Všechny elementy `<span>`, které obsahují text položky stromu mají atribut `id`, který se skládá z písmene "b" a čísla představujícího id boxu ve stromu. Hodnota id se získává pomocí metody `getId` třídy `Box`. Tato identifikace umožňuje provázání elementů ve stromu a v renderované stránce vytvořené na základě stromu boxů. V samotném textu jsou speciální HTML znaky (např. levá závorka, pravá závorka apod.) nahrazené odpovídajícími znakovými HTML entitami pomocí procedury `escapeHtml` balíku `org.apache.commons`.

```
tree += "<span id='b" + node.getId() + "'>" +
escapeHtml(node.toString()) + "</span>";
```

### 4.3.4 Vytvoření HTML kódu stromu oblastí

HTML kód stromu oblastí se vytváří rekurzivně pomocí metody `htmlAreaTree`. Vstupním parametrem této metody je objekt typu `Area`, představující uzel stromu oblastí. Na začátku se metodě jako parametr předává kořenový uzel a následně se prochází všichni jeho potomci. Pokud jeden z potomků je dalším uzlem, začnou se rekurzivně procházet jeho potomci, dokud neprojdeme všechny uzly stromu a koncové elementy. Struktura stromu je popsaná v podkapitole Realizace stromů.

Všechny elementy `<span>`, které obsahují text položky stromu mají atribut `id`, který se skládá z písmene "a" a čísla představujícího id oblasti ve stromu. Hodnota id se získává pomocí metody `getId` třídy `Area`. Tato identifikace umožňuje provázání elementů ve stromu a v renderované stránce vytvořené na základě stromu oblastí. V samotném textu jsou speciální HTML znaky (např.

levá závorka, pravá závorka apod.) nahrazené odpovídajícími znakovými HTML entitami pomocí procedury `escapeHtml` balíku `org.apache.commons`.

```
tree += "<span id='a" + node.getId() + "'>" +
escapeHtml(node.toString()) + "</span>";
```

### 4.3.5 Informace o vlastnostech oblastí

Pro získání vlastností oblasti se používá metoda `getAreaInfo`, která dostává jako vstupní parametr objekt třídy `Area`. Tato metoda slouží k vytvoření HTML kódu pro zobrazení informací vybrané oblasti ve stromu oblastí. HTML formát prezentace vlastností je popsán v podkapitole Zobrazení vlastností oblastí.

Počet a typy vlastností a také postup jejich získání odpovídají implementaci dle FITLayout Framework. Výsledkem je objekt třídy `Map` obsahující páry: název vlastnosti (key) a hodnota vlastnosti (value). Poté je z tohoto objektu vytvořen HTML kód vlastností v podobě HTML tabulky.

### 4.3.6 Pomocné metody

Metoda `getAreaInfo` používá dvě pomocné metody:

- `borderString` - vrací řetězec znázorňující ohraničení vybrané oblasti (zda má horní, spodní, levé nebo pravé ohraničení)
- `colorString` - vrací barvu pozadí oblasti ve formátu RGB (dvě hexadecimální číslice na každou barvu) a nebo "transparent" pokud je oblast průhledná

Formát návratových hodnot těchto metod odpovídá implementaci dle FITLayout Framework.

# 5. Testování

## 5.1 Testování vytvořeného řešení a porovnání funkčnosti s FITLayout Framework

Předmětem testování byla správnost provedení funkcí vyjmenovaných v kapitole Seznam realizovaných funkcí a porovnání výsledků testované dynamické stránky a FITLayout Framework. Postup testování a jeho výsledky jsou uvedeny v Tabulce 5.1.

Testování webové aplikace se provádělo za následujících podmínek:

- spuštěný Eclipse Java EE IDE for Web Developers, ver. Kepler Service Release 2
- spuštěný Apache Tomcat 7.0
- spuštěný výchozí webový prohlížeč Google Chrome verze 50.0.2661.94 m

| **Testovaná funkce / postup testování** | **Výsledky testování** | |
|---|---|---|
| | **Aplikace FITLayout** | **FITLayout Framework** |
| **1. Zadání adresy www stránky** | | |
| zadání správné adresy | Zobrazila se požadovaná stránka | Zobrazila se požadovaná stránka |
| zadání nesprávného tvaru url adresy | Zobrazilo se hlášení "Zadejte prosím adresu URL" | Program neprovedl operaci. Žádné hlášení se nezobrazilo |
| zadání neexistující adresy | Stránka se nezobrazila (Server could not provide requested page) | Program neprovedl operaci. Žádné hlášení se nezobrazilo |
| **2. Nastavení šířky renderované stránky** | | |
| ponechání výchozí hodnoty (1200 pixelů) | Zobrazila se stránka o šířce 1200 pixelů | Zobrazila se stránka o šířce 1200 pixelů |
| zadání hodnoty menší než je minimální povolená hodnota | Zobrazilo se hlášení "Hodnota musí být vetší nebo rovna 768" | Nebylo zjištěno žádné nastavení minimální hodnoty |
| zadání nečíselné hodnoty | Aplikace neumožňuje vepsat do pole žádné nečíselné hodnoty | Nebylo možné nastavit nečíselnou hodnotu. Hodnota v poli se nastavila na poslední použitou číselnou hodnotu |
| zadání jiné číselné hodnoty | Zobrazila se stránka v požadované šířce | Zobrazila se stránka v požadované šířce. Program neumožnil zadat hodnotu vyšší než 9999 |
| **3. Nastavení výšky renderované stránky** | | |
| ponechání výchozí hodnoty (800 pixelů) | Zobrazila se stránka o výšce 800 pixelů | Zobrazila se stránka o výšce 800 pixelů |
| zadání hodnoty menší než je minimální povolená hodnota | Zobrazilo se hlášení "Hodnota musí být vetší | Nebylo zjištěno žádné nastavení minimální |

| Testovaná funkce / postup testování | | Výsledky testování | |
|---|---|---|---|
| | | **Aplikace FITLayout** | **FITLayout Framework** |
| | | nebo rovna 360" | hodnoty |
| zadání nečíselné hodnoty | | Aplikace neumožňuje vepsat do pole žádné nečíselné hodnoty | Nebylo možné nastavit nečíselnou hodnotu. Hodnota v poli se nastavila na poslední použitou číselnou hodnotu |
| zadání jiné číselné hodnoty | | Zobrazila se stránka v požadované výšce | Zobrazila se stránka v požadované výšce. Program neumožnil zadat hodnotu vyšší než 9999 |
| **4.** | **Výběr rendereru** | Kliknutím na rozbalovací pole bylo možné vybrat renderer | Kliknutím na rozbalovací pole bylo možné vybrat renderer |
| **5.** | **Spuštění rendereru** | | |
| zmáčknutí tlačítka `Go!` | | Zobrazila se zadaná stránka a strom boxů | Zobrazila se zadaná stránka a strom boxů |
| **6.** | **Výběr segmentátoru** | Kliknutím na rozbalovací pole bylo možné vybrat segmentátor | Kliknutím na rozbalovací pole bylo možné vybrat segmentátor |
| **7.** | **Spuštění segmentátoru** | | |
| zmáčknutí tlačítka `Run` | | Zobrazila se zadaná stránka a strom vizuálních oblastí | Zobrazila se zadaná stránka a strom vizuálních oblastí |
| zaškrtnutí políčka `Run automatically` a zmáčknutí tlačítka `Go!` | | Zobrazila se zadaná stránka a strom vizuálních oblastí | Zobrazila se zadaná stránka a strom vizuálních oblastí |
| **8.** | **Vizualizace renderované webové stránky** | | |
| výchozí stav | | Prostřední okno je prázdné | V prostředním okně je zobrazená stránka výchozí url adresa (`http://cssbox.sf.net`) |
| po spuštění | | Stránka se zobrazila v prostředním okně podle zadaných parametrů | Stránka se zobrazila v prostředním okně podle zadaných parametrů |
| **9.** | **Generování stromů oblastí a boxů** | | |
| Strom boxů | | Strom boxů se vygeneroval po zmáčknutí tlačítka `Go!` a nebo `Run` | Strom boxů se vygeneroval po zmáčknutí tlačítka `Go!` a nebo `Run` |
| Strom oblastí | | Strom oblastí se vygeneroval po zmáčknutí tlačítka `Run` nebo zaškrtnutí pole `Run Automatically` a | Strom oblastí se vygeneroval po zmáčknutí tlačítka `Run` nebo zaškrtnutí pole `Run Automatically` a |

| Testovaná funkce / postup testování | | Výsledky testování | |
|---|---|---|---|
| | | **Aplikace FITLayout** | **FITLayout Framework** |
| | | zmáčknutí tlačítka `Go!` | zmáčknutí tlačítka `Go!` |
| **10.** | **Přepínání mezi označení oblastí (`Area`) a boxů (`Box`)** | | |
| výchozí nastavení | | Tlačítko `Area` je označené jako aktivní a tlačítko `Box` jako neaktivní | Tlačítko `Area` je označené jako aktivní a tlačítko `Box` jako neaktivní |
| zmáčknutí tlačítka `Box` | | Tlačítko `Box` se označilo jako aktivní a tlačítko `Area` jako neaktivní | Tlačítko `Box` se označilo jako aktivní a tlačítko `Area` jako neaktivní |
| zmáčknutí tlačítka `Area` | | Tlačítko `Area` se označilo jako aktivní a tlačítko `Box` jako neaktivní | Tlačítko `Area` se označilo jako aktivní a tlačítko `Box` jako neaktivní |
| **11.** | **Označení boxů a oblastí** | | |
| tlačítko `Box` je aktivní | | Kliknutím na stránku se zelenou barvou ohraničil vybraný box na stránce a zobrazil se ve stromu boxů jako aktivní | Kliknutím na stránku se zelenou barvou ohraničil vybraný box na stránce a zobrazil se ve stromu boxů jako aktivní |
| tlačítko `Area` je aktivní | | Kliknutím na stránku se fuchsiovou barvou ohraničila vybraná oblast na stránce a zobrazila se ve stromu oblastí jako aktivní | Kliknutím na stránku se fuchsiovou barvou ohraničila vybraná oblast na stránce a zobrazila se ve stromu oblastí jako aktivní |
| **12.** | **Odstranění označení boxů a oblastí** | | |
| zmáčknutí tlačítka `Clear` | | Zmizelo veškeré barevné označení na zobrazené stránce | Zmizelo veškeré barevné označení na zobrazené stránce |
| **13.** | **Výběr aktivního stromu** | | |
| výchozí nastavení | | Aktivní je záložka `Box Tree` | Aktivní je záložka `Area Tree` |
| kliknutí na záložku `Box Tree` | | Záložka se stala aktivní `Box Tree` | Záložka se stala aktivní `Box Tree` |
| kliknutí na záložku `Area Tree` | | Záložka se stala aktivní `Area Tree` | Záložka se stala aktivní `Area Tree` |
| **14.** | **Označení položky stromu na stránce** | | |
| kliknutí na položku v `Box Tree` | | Na stránce se zelenou barvou ohraničil vybraný box a položka v `Box Tree` se označila | Na stránce se zelenou barvou ohraničil vybraný box a položka v `Box Tree` se označila |
| kliknutí na položku v `Area Tree` | | Na stránce se fuchsiovou barvou ohraničila vybraná oblast a položka v `Area` | Na stránce se fuchsiovou barvou ohraničila vybraná oblast a položka v `Area` |

| Testovaná funkce / postup testování | | Výsledky testování | |
|---|---|---|---|
| | | **Aplikace FITLayout** | **FITLayout Framework** |
| | | `Tree` se označila | `Tree` se označila |
| **15.** | **Výpis vlastností oblasti** | | |
| kliknutí na položku v `Area Tree` | | V tabulce se zobrazily vlastnosti zvolené položky | V tabulce se zobrazily vlastnosti zvolené položky |
| kliknutí na položku v `Box Tree` | | Tabulka zůstala beze změny | Tabulka zůstala beze změny |
| **16.** | **Obnovení výchozího stavu stromu** | | |
| zmáčknutí tlačítka `Refresh` | | Uzly obou stromů se sbalily a otevřené zůstaly pouze kořenové uzly | Uzly obou stromů se sbalily a otevřené zůstaly pouze kořenové uzly |
| **17.** | **Označení boxu vybraného elementu ve stromu boxů a boxů všech jeho potomků** | | |
| kliknutí na tlačítko `Show Boxes` | | Zelenou barvou se ohraničili všichni potomci označeného boxu ve stromu boxů | Zelenou barvou se ohraničili všichni potomci označeného boxu ve stromu boxů |
| **18.** | **Označení oblasti vybraného elementu ve stromu oblastí a oblastí všech jeho potomků** | | |
| kliknutí na tlačítko `Show Areas` | | Fuchsiovou barvou se ohraničili všichni potomci označené oblasti ve stromu oblastí | Fuchsiovou barvou se ohraničili všichni potomci označené oblasti ve stromu oblastí |

*Tabulka 5.1 Testování vytvořeného řešení a porovnání funkčnosti s FITLayout Framework*

Aplikace FITLayout Framework má více funkcí a ovládacích prvků uživatelského rozhraní, zejména vytváření stromu logických oblastí a správu operátorů. Logické oblasti ještě nemají konečnou podobu, a proto se ve webové aplikaci neimplementovaly. Správa operátorů je do velké míry vázaná na rozhraní Java Swing a její implementace na webové stránce vyžaduje změny v samotném FITLayout Framework. Proto tato implementace také nebyla provedena.

Uvedený postup testování byl aplikován také pro prohlížeče Mozilla Firefox 45.0.1, Internet Explorer 11 a Opera 37.0 se stejnými výsledky.

# 6. Závěr

Cílem této práce bylo navrhnout a poté implementovat webovou aplikaci pro analýzu internetových dokumentů. Návrh funkcí této aplikace vycházel z funkcí již existující desktopové aplikace FITLayout Framework. Implementaci předcházelo nastudování potřebných technologií pro realizaci dynamických webových stránek, které jsou popsány v kapitole 2. Klientská část aplikace byla implementovaná pomocí jazyka HTML, kaskádových stylů CSS a knihovny Bootstrap, dynamického jazyka JavaScript s použitím knihovny jQuery. Serverová část aplikace byla implementovaná na

platformě Java EE a jako aplikační server byl použit Apache Tomcat. Celá aplikace byla vyvíjená a laděna ve vývojovém prostředí Eclipse Java EE IDE for Web Developers, ver. Kepler Service Release 2.

V závěru bylo provedeno testování funkcí webové aplikace a výsledky testování byly porovnány s funkcemi FITLayout Framework. Tyto výsledky jsou uvedeny v kapitole 5.1 v Tabulka 5.1. Výsledky testování ukázaly, že cíle práce byly dosažené, webová aplikace plní svoji funkci v souladu s návrhem řešení a tyto funkce se shodují s funkcemi desktopové aplikace FITLayout Framework.

Je nutné podotknout, že desktopová aplikace FITLayout Framework je stále ve vývoji, proto nebylo možné implementovat všechny funkce a je možné, že se i webová aplikace v budoucnu rozšíří o další funkce.

# Literatura


**[1]** **Burget, Radek.** FITLayout Framework Manual. [Online] 2016. [Citace: 30. duben 2016.] http://www.fit.vutbr.cz/~burgetr/FITLayout/manual/.

**[2]** **Burget, Radek.** Základy jazyka HTML. *Tvorba webových stránek - Přednášky.* [Online] 17. únor 2015. [Citace: 15. duben 2016.] https://www.fit.vutbr.cz/study/courses/ITW/private/prednasky/itw_p02.pdf.

**[3]** Основы HTML. *HTML5BOOK.RU - HTML, CSS, JavaScript и jQuery.* [Online] 2016. [Citace: 20. duben 2016.] http://html5book.ru/osnovy-html/#part1.

**[4]** **Janovský, Dušan.** Doctype deklarace. *Jak psát web o tvorbě, údržbě a zlepšování internetových stránek.* [Online] 2016. [Citace: 15. duben 2016.] http://www.jakpsatweb.cz/doctype.html. ISSN 1801-0458.

**[5]** HTML5 Introduction. *W3Schools Online Web Tutorials.* [Online] 2016. [Citace: 3. květen 2016.] http://www.w3schools.com/html/html5_intro.asp.

**[6]** **Burget, Radek.** Úvod do kaskádových stylů (CSS). *Tvorba webových stránek - Přednášky.* [Online] 3. březen 2015. [Citace: 15. duben 2016.] https://www.fit.vutbr.cz/study/courses/ITW/private/prednasky/itw_p03.pdf.

**[7]** Bootstrap (фреймворк) - Википедия. *Википедия - свободная энциклопедия.* [Online] 20. duben 2016. [Citace: 3. květen 2016.] https://ru.wikipedia.org/wiki/Bootstrap_(фреймворк).

**[8]** **Burget, Radek.** JavaScript a jQuery. *Tvorba webových stránek - Přednášky.* [Online] 14. duben 2015. [Citace: 25. duben 2016.] https://www.fit.vutbr.cz/study/courses/ITW/private/prednasky/itw_p09.pdf.

**[9]** Основы JavaScript. *HTML5BOOK.RU - HTML, CSS, JavaScript и jQuery.* [Online] 2016. [Citace: 15. duben 2016.] http://html5book.ru/osnovy-javascript/.

**[10]** Введение в jQuery. *HTML5BOOK.RU - HTML, CSS, JavaScript и jQuery.* [Online] 2016. [Citace: 15. duben 2016.] http://html5book.ru/vvedenie-v-jquery/.

**[11]** **Hanel, David.** Kapitola 2. Základní principy Java Enterprise Edition. *Vývoj webových aplikací pomocí frameworku JavaServer Faces.* [Online] prosinec 2009. [Citace: 20. duben 2016.] http://java.vse.cz/jsf/chunks/ch02.html.

**[12]** **Matoušek Petr**. Kapitola 2. Programování sítí TCP/IP. *Síťové služby a jejich architektura*. Brno: Nakladatelství Vysokého učení technického v Brně VUTIUM, 2014. [Citace: 5. květen 2016.] ISBN 978-80-214-3766-1

**[13]** **Hanel, David.** 2.2. Aplikační server. *Vývoj webových aplikací pomocí frameworku JavaServer Faces.* [Online] prosinec 2009. [Citace: 20. duben 2016.] http://java.vse.cz/jsf/chunks/ch02s02.html.

**[14]** Eclipse (software). *Wikipedia, the free encyclopedia.* [Online] 8. květen 2016. [Citace: 8. květen 2016.] https://en.wikipedia.org/wiki/Eclipse_(software).

# Seznam obrázků



# Seznam tabulek



# Seznam zkratek

FIT - Fakulta informačních technologií
VUT - Vysoké učení technické
HTML - HyperText Markup Language
CSS - Cascading Style Sheets
DTD - Document Type Definition
CMS - Content Management System
DOM - Document Object Model
AJAX - Asynchronous JavaScript and XML
XML - Extensible Markup Language
API - Application Programming Interface
JSP - JavaServer Pages
JSF - JavaServer Faces
SWT - Standard Widget Toolkit
RCP - Rich Client Platform

# Obsah přiloženého CD

Příloha 1. Bakalářská práce-Olga Mirská.pdf
Příloha 2. Bakalářská práce-Olga Mirská.docx
Příloha 3. Manuál.pdf
Příloha 4. Zdrojové soubory
Příloha 5. FITLayout.zip
Příloha 6. readme.txt

# Příloha A. Uživatelský manuál

# 1. Požadavky pro spuštění programu

## 1.1.Minimální požadavky na hardware

Pro spuštění programu je třeba následující hardware:

- 512 MB paměti RAM
- 1GHz procesor
- 800 MB prostoru na pevném disku
- grafická karta 2 mil. barev
- monitor s velikostí obrazovky 1024x768

## 1.2 Minimální požadavky na software

Pro spuštění programu je třeba mít nainstalovaný následující software:

- Eclipse Java EE IDE for Web Developers, ver. Kepler Service Release 2
- Java EE 1.7
- Apache Tomcat 7.0
- webový prohlížeč (Google Chrome 50.x, IE 8, Mozilla Firefox 42.x nebo vyšší)

# 2. Instalace webové aplikace

## 2.1 Nastavení Tomcat v Eclipse

Nastavení Tomcat v Eclipse proveďte následovně:

- Spusťte Eclipse
- Vyberte položku menu Windows -> Preferences. Otevře se dialogové okno Preferences.
- Vyberte položku Server -> Runtime Environments
- Pomocí tlačítka Add přidejte Apache Tomcat 7.0

## 2.2 Instalace webové aplikace FITLayout

Instalaci webové aplikace proveďte následovně:

- Vyberte položku menu File -> Import
- Vyberte General -> Existing Project into Workspace a zmáčkněte tlačítko Next
- Vyberte Select Archive File, zmáčkněte tlačítko Browse a vyberte cestu k souboru FITLayout.zip
- Označte v okně Projects položku FITLayout
- Zmáčkněte tlačítko Finish

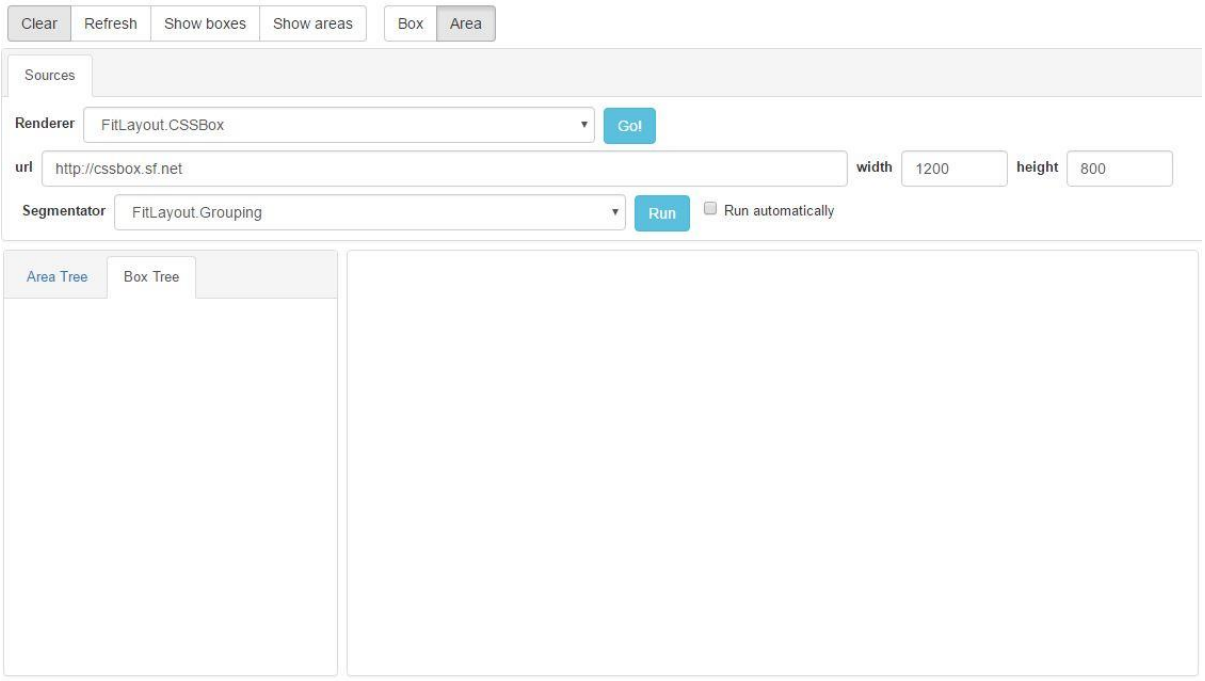
## 2.3 Přidat webovou aplikaci do Tomcat

Webová aplikace se přidá do Tomcat následujícím způsobem:
Vyberte položku menu Window -> Show View -> Servers. Ve spodní části se zobrazí záložka s položkou Tomcat.
Klikněte pravým tlačítkem na položku Tomcat. Vyberte Add and Remove
Vyberte v levém okně položku FITLayout, zmáčkněte tlačítko Add a zmáčkněte tlačítko Finish

# 3. Běh webové aplikace

## 3.1 Spuštění aplikace

V projektu FITLayout vyberte složku WebContent -> index.jsp a spusťte jej. Ve výchozím webovém prohlížeči se otevře okno aplikace.




*Obrázek 3.1 Okno aplikace FITLayout*

## 3.2 Ovládací prvky aplikace

**3.2.1 Tlačítko Clear** - odstraňuje barevné ohraničení boxů a oblastí
**3.2.2 Tlačítko Refresh** - uvádí stromy boxu a oblastí do výchozího stavu
**3.2.3 Tlačítko Show Boxes** - označuje na stránce box vybraného elementu ve stromu boxu a také boxy všech jeho potomků
**3.2.4 Tlačítko Show Areas** - označuje na stránce oblast vybraného elementu ve stromu oblastí a také oblasti všech jeho potomků
**3.2.5 Tlačítko Box** - umožňuje označit boxy na stránce pomocí kliknutí myši
**3.2.6 Tlačítko Area** - umožňuje označit oblasti na stránce pomocí kliknutí myši
**3.2.7 Renderer** - výběr názvu rendereru
**3.2.8 Tlačítko Go!** - spuštění renderování stránky
**3.2.9 Pole url** - zadání adresy požadované webové stránky
**3.2.10 Pole width** - zadání šířky renderované stránky
**3.2.11 Pole height** - zadání výšky renderované stránky
**3.2.12 Segmentator** - výběr názvu segmentátoru
**3.2.13 Tlačítko Run** - spuštění segmentace a renderování
**3.2.14 Run Automatically** - pokud je pole zaškrtnuté, spolu s renderováním se spustí i segmentace
**3.2.15 Záložka Area Tree** - zobrazí se strom oblastí
**3.2.16 Záložka Box Tree** - zobrazí se strom boxů

## 3.3 Výběr oblasti/boxů

Oblast/box se označí na stránce kliknutím levým tlačítkem myši na položku odpovídajícího stromu. Při kliknutím levým tlačítkem myši na renderovanou stránku se označí box nebo oblast dle aktivního tlačítka Box nebo Area.